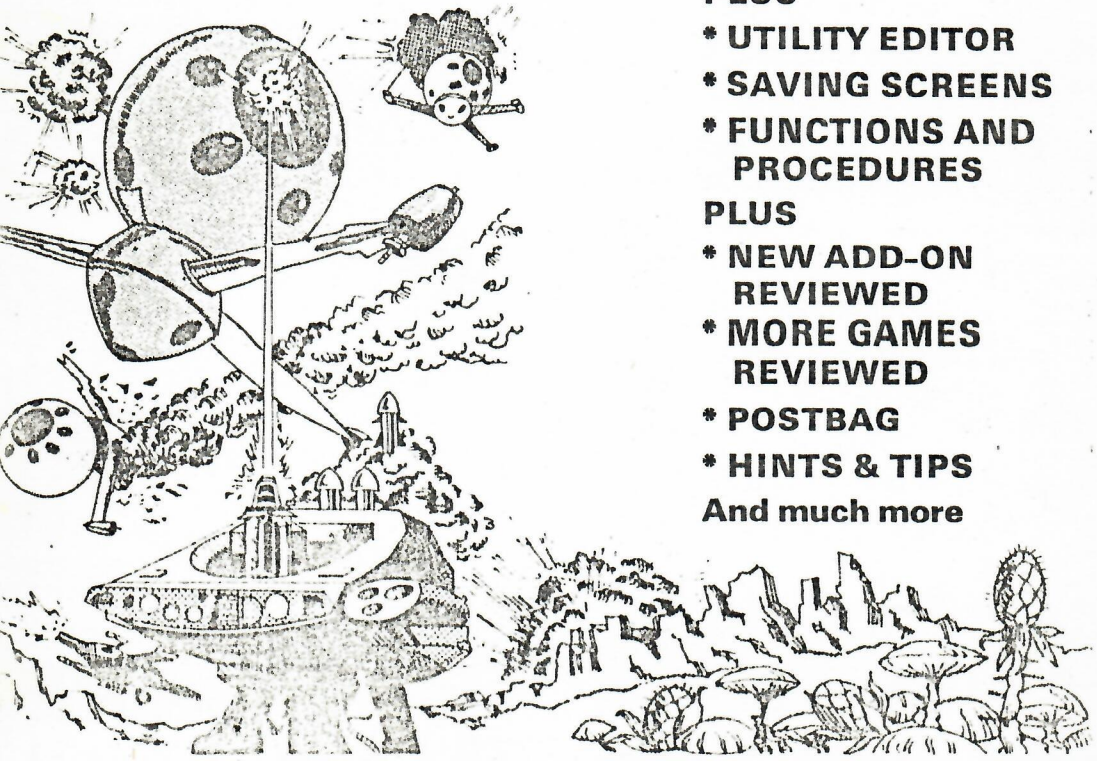
£1.00

ELBUG FOR THE ELECTRON

Vol 1 No 5 APRIL 1984

INVASION OF THE ALIENS



GAMES

- **ELEVATOR MANIA**
- **DOMINOES**

PLUS

- **UTILITY EDITOR**
- **SAVING SCREENS**
- **FUNCTIONS AND PROCEDURES**

PLUS

- **NEW ADD-ON REVIEWED**
- **MORE GAMES REVIEWED**
- **POSTBAG**
- **HINTS & TIPS**

And much more

EDITORIAL

THIS MONTH'S MAGAZINE

If you are interested in programming, then you should find the Utility Editor in this issue of considerable help and interest. The program includes facilities to search for, or to search for and replace, any string of characters in your own program, including Basic keywords, very useful, for example, if you want to change the name of a variable throughout your program, and it will also list out all your functions and procedures and tell you where they are.

And talking of functions and procedures, if these are something that you are still not quite sure about, the article by Rob Pickering and Philip Le Grand will really help you on your way. The article also tells you how you can build up a library of your own most useful routines.

Your first library routine could well be the procedure described in another of this month's articles which allows you to save to cassette any screen display, such as those that you can produce using ASTAAD, our computer aided design program that we published in issue No.3. The article also tells you how to re-display any screen image that you have previously saved to cassette.

If you read the article on saving screens, you will see that we are running a competition with a prize for the best screen display sent to us. The prize is being donated by SIR Computers of Cardiff, and will be one of their new add-ons for the Electron. This product, a ROM expansion board, is now available, and you can read all about it in this month's review.

We have also just heard that the second add-on from SIR Computers is now available, an interface for printer and joysticks, which we shall also be reviewing.

There are many more articles, programs and reviews, including three more excellent games for your enjoyment, making this, without doubt, one of the best issues of ELBUG yet.

POSTBAG

I said last month how much we enjoy hearing from you, particularly if you have any ideas of articles or programs you would like to see appearing in the magazine. You also continue to be very complimentary about the magazine and this includes one reader who was so pleased that he asked if we could produce the magazine every week, and not just every month! Phew!

ADVERTISING SUPPLEMENT

As you may know the advertising supplement is sent out to readers of both ELBUG and BEEBUG, our highly successful magazine for users of the BBC micro. All the time, we are taking steps to make this more useful and interesting to ELBUG readers, including our own Software Club offers. Do remember that many of our advertisers, and this includes dealers listed in our Readers' Discounts pages, do cater for Electron users and will help ELBUG members.

Mike Williams

NOTICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BOA

In response to requests from ELBUG members, we have decided to make available each month, a cassette containing all the programs from that month's magazine. We will be giving all the details of this, and of cassettes for all back copies, in the next issue of ELBUG.

We have now installed a telephone line to deal with any enquiries you may have relating to magazine subscriptions, orders for backcopies, software etc. It will not be possible to answer technical queries over the telephone. The number to ring is St Albans 60263 and the line will be manned from 1pm to 4pm each weekday.

ELBUG MAGAZINE

GENERAL CONTENTS

Page Contents

2 Editorial
4 Invasion of The Aliens
6 Electron Graphics (Part 5)
9 Postbag
10 Saving and Loading Screens
12 New Electron Add-on Reviewed
14 Using Procedures and Functions
17 Fabric Patterns
18 Dominoes
22 Utility Editor
26 Latest Games Reviewed
28 Using BBC Micro Programs on an Electron (Part 2)
30 Elevator Mania

HINTS & TIPS

Page Contents

11 Easier Entry of Lower Case Variable Names
11 Random Events
11 Two Key Entry for ENDPROC
11 Easier Viewing of Text
13 Transparent Key Definitions
29 Time Delay
34 Freezing a Program in Basic

PROGRAMS

Page Contents

4 Invasion of The Aliens (game)
6 Electron Graphics Examples
10 Saving and Loading Screen Displays
17 Fabric Patterns
18 Dominoes (game)
24 Utility Editor
30 Elevator Mania (game)

INVASION OF THE ALIENS

by Alan Webster

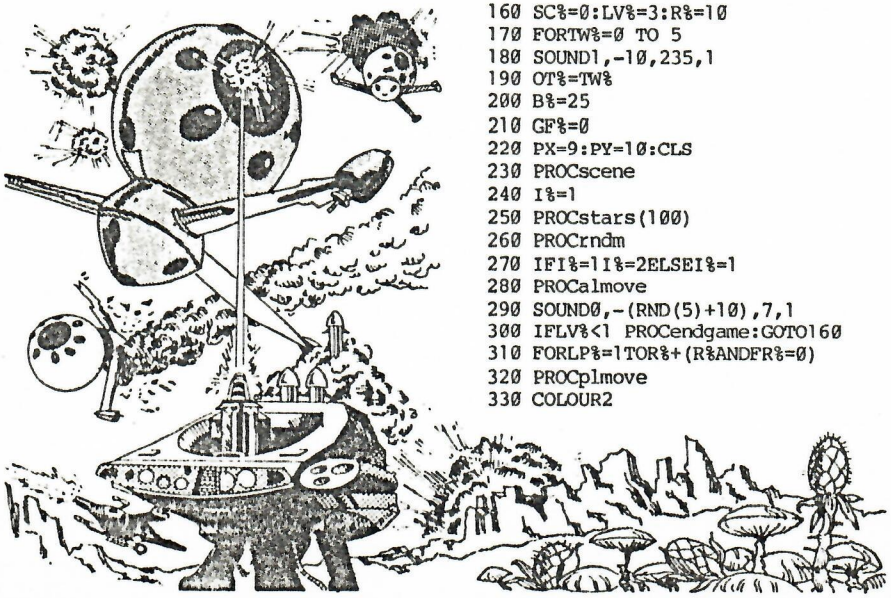
This is a space invader type game for one player in which the object is to destroy all of the descending aliens (isn't it always?). The game starts with one alien attempting to 'land' on the surface of the planet. If you succeed in zapping it, then you move onto the next sheet with two aliens. This goes on until you have beaten off the sheet with six of the creatures, whereupon the game restarts with just one alien, but becomes faster...and faster...and faster...!

You have three lives to play with and you lose one every time an alien lands. After losing all of your lives your score is displayed and the highest score shown.

Your weapon is aimed by moving a cross-sight until it is centred on an invader. The keys to use are as follows:

Z	Left	:	Up
X	Right	/	Down

Space Bar to Fire

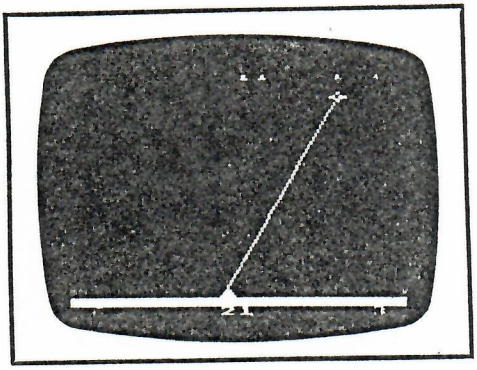


As you line up your sight, the alien erects a temporary psychokinetic force field which hides it from view, though without creating any protection from your missile.

You score 1 point for every alien hit on the first wave (sheets 1 to 6), two points on the second wave and so on. You are also limited to the number of missiles you can fire and this is displayed at the bottom centre of the screen and is given in line 200. It is set to 25 in the program listed, but of course you can change this to make the game easier or harder.

```
 10 REM PROGRAM INVASION
 20 REM AUTHOR  Alan R. Webster
 30 REM VERSION E0.1
 40 REM ELBUG   APRIL 1984
 50 REM PROGRAM SUBJECT TO COPYRIGHT
 60:
100 ON ERROR GOTO 1710
110 MODE5
120 EK%=0:FR%=0
130 VDU19,1,11,0,0,0
140 PROCchr
150 HI%=0:DIM Z(5,2)
160 SC%=0:LV%=3:R%=10
170 FORTW%=0 TO 5
180 SOUND1,-10,235,1
190 OT%=TW%
200 B%=25
210 GF%=0
220 PX=9:PY=10:CLS
230 PROCscene
240 I%=1
250 PROCstars(100)
260 PROCrndm
270 IFI%=1I%=2ELSEI%=1
280 PROCalmove
290 SOUND0,-(RND(5)+10),7,1
300 IFLV%<1 PROCendgame:GOTO160
310 FORLP%=1TOR%+(R%ANDFR%=0)
320 PROCplmove
330 COLOUR2
```

```
 340 PRINTTAB(2,31);SC%;TAB(17,31);LV%
;
 350 COLOUR3
 360 PROCfire
 370 NEXTLP%
 380 IFEK%=TW%+1 EK%=0:NEXTTW%
 390 IF TW%=6 R%=R%-2:GOTO170 ELSE270
 400 END
 410:
1000 DEFPROCscene
1010 VDU23,1,0;0;0;0;
1020 VDU23,255,255,255,255,255,255,255
,255,255
1030 FORA%=1TO18:PRINTTAB(A%,30);CHR$2
55:NEXT
1040 VDU23,254,8,8,28,28,62,62,127,127
1050 PRINTTAB(9,29);CHR$254
1060 ENDPROC
1070:
1080 DEFPROCstars(A%)
1090 FORA1%=1TOA%:GCOL0,RND(3)
1100 PLOT69,RND(1280)-1,RND(900)+123
1110 NEXT:ENDPROC
```



```
1120:
1130 DEFPROCchr
1140 VDU23,224,24,24,36,231,231,36,24,
24
1150 VDU23,225,102,153,60,90,126,60,66
,129
1160 VDU23,226,129,90,60,90,126,60,66,
36
1170 ENDPROC
1180:
1190 DEFPROCrndm
1200 FORA%=0TOTW%:Z(A%,0)=RND(18):Z(A%
,1)=0:Z(A%,2)=1:NEXT
1210 ENDPROC
1220:
1230 DEFPROCplmove
1240 OX=PX:OY=PY
1250 IFINKEY-98ANDPX>1GF%=GF%-1
1260 IFINKEY-67ANDPX<18GF%=GF%+1
1270 IF ABS(GF%)=2 PX=PX+SGN(GF%):GF%=
0
1280 IFINKEY-105ANDPY<27PY=PY+1
1290 IFINKEY-73ANDPY>0PY=PY-1
1300 PRINTTAB(OX,OY);" ":COLOUR3:PRINT
TAB(PX,PY)CHR$224
1310 ENDPROC
1320:
1330 DEFPROCalmove
1340 FORA%=0TOTW%
1350 IFZ(A%,2)=0 GOTO 1440
1360 PRINTTAB(Z(A%,0),Z(A%,1));SPC1
1370 Z(A%,1)=Z(A%,1)+1:Z(A%,0)=Z(A%,0)
+(RND(3)-2)
1380 G%=Z(A%,0)
1390 IFG%<1Z(A%,0)=1
1400 IFG%>18Z(A%,0)=18
1410 IFZ(A%,1)=29 AND G%=3ORG%=9ORG%=1
6 Z(A%,0)=Z(A%,0)+1
1420 COLOUR1:PRINTTAB(Z(A%,0),Z(A%,1))
CHR$(224+I%)
1430 IFZ(A%,1)=29 Z(A%,2)=0:LV%=LV%-1:
EK%=EK%+1:FORSD%=1TO2:SOUND1,-15,5,4:SO
UND1,-15,1,4:NEXT
1440 NEXT
1450 ENDPROC
1460:
1470 DEFPROCendgame
1480 A%=INKEY(310)
1490 CLS
1500 PRINTTAB(3,15);"YOUR SCORE:";SC%
1510 IFSC%>HI%HI%=SC%
1520 PRINTTAB(3,17);"HI SCORE  :";HI%
1530 *FX15,0
1540 COLOUR3:EK%=0
1550 S=GET
1560 ENDPROC
1570:
1580 DEFPROCfire
1590 FR%=0
1600 IFINKEY-99ANDB%>0FR%=608
1610 IFFR%=0 ENDPROC
1620 FX=64*PX+32:FY=1024-(32*PY)-16
1630 FR1%=96
1640 MOVEFR%,FR1%:DRAWFX,FY:GCOL0,0
1650 FORY%=0TOTW%:IFPX=Z(Y%,0)ANDPY=Z(
Y%,1)ANDZ(Y%,2)<>0 Z(Y%,2)=0:SC%=SC%+11
-R%:EK%=EK%+1:SOUND1,-15,100,1:SOUND1,-
14,2,1:SOUND1,-15,200,1 ELSE SOUND1,-10
,235,1
1660 NEXT
1670 MOVEFR%,FR1%:DRAWFX,FY:GCOL0,3
1680 B%=B%-1:PRINTTAB(9,31);B%;CHR$32;
1690 ENDPROC
1700:
1710 ON ERROR OFF
1720 MODE6
1730 IF ERR=17 END
1740 REPORT:PRINT" at line ";ERL
1750 END
```

ELECTRON GRAPHICS (Part 5)

by Mike Williams

We continue our introductory series on Electron Graphics by looking this month at the use of the PLOT command, the most comprehensive of the various graphics commands available on the Electron.

Last month we looked at the MOVE and DRAW commands, and their use in drawing a variety of open geometrical shapes (triangle, rectangle, circle, polygon). In most of these examples we wrote short and flexible procedures for the task in hand, something we shall continue to do this month.

THE PLOT COMMAND

The PLOT command, available in BBC Basic, is a general graphics command having many different uses. The format of the command is always the same:

```
PLOT k, x, y
```

The value of k, which can nominally be any value in the range 0 to 255, determines the exact function carried out by the PLOT command. The values of x and y refer to a point on the graphics area of the screen as explained last month (1280 points horizontally, 1024 points vertically).

FILLING TRIANGLES AND RECTANGLES

The first, and one of the most useful applications of the PLOT command is in producing filled shapes on the screen. The command:

```
PLOT 85, X, Y
```

will plot and fill a triangle on the screen using the drawing colour currently set by the GCOL command (or white by default). The three vertices (or corners) of the triangle are taken to be the last two points previously specified and the point x, y given in the PLOT command itself. For example try the following short program:

```
100 MODE 2
110 GCOL 0,3
120 MOVE 200,100
130 MOVE 640,800
140 PLOT85,1080,100
150 END
```

The program is written in Mode 2 and selects yellow (colour 3) as the drawing colour. Using two MOVE commands and one PLOT 85 command, a yellow filled triangle is then drawn using (200, 100), (640, 800) and (1080, 100) as the three vertices. You will notice that the two sloping sides of the triangle have a stepped appearance. This appearance depends on the resolution being used (low resolution in Mode 2). If you try the same program using Mode 1 (medium resolution) and Mode 0 (high resolution) you will be able to see the effect that this has. You may also wish to refer back to last month's article for more information on resolution.

A filled triangle is the only shape that is immediately available to us, and so any other filled shape must usually be constructed from triangles. Consider, for example the task of drawing a filled rectangle.

Suppose we want a green rectangle, 400 units high by 600 long and with the bottom left hand corner in position (200, 100). It is worth just trying to work out the sequence of commands for yourself, and to see how few you really do need. A rectangle is, of course, made up from two triangles and the shortest way of achieving this is as follows:

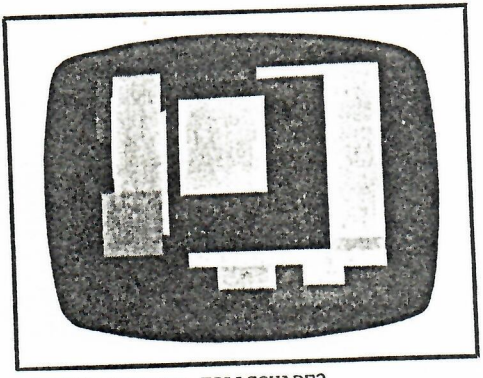
```
100 MODE 2
110 GCOL 0,1
120 MOVE 200,100: MOVE 200,500
130 PLOT 85,800,100:PLOT 85,800,500
140 END
```

Notice how, by moving from one vertex to the next in the right order, we can draw our rectangle with just four commands. Basic always remembers the co-ordinates of the last two points visited.

We can rewrite this as a procedure which will display a square of any size, and in any colour, anywhere on the screen. The procedure, PROCfillsquare, is defined as follows:

```
100 MODE 2
110 ON ERROR GOTO 190
120 REPEAT
130 colour=RND(7):size=RND(1000)
140 x=RND(1279): y=RND(1023)
150 PROCfillsquare(colour,size,x,y)
160 UNTIL FALSE
170 END
180 :
190 ON ERROR OFF:MODE 6
200 REPORT:PRINT" at line ";ERL:END
210 :
1000 DEF PROCfillsquare(colour,size,x,
y)
1010 GCOL0,colour
1020 MOVE x,y: MOVE x,y+size
1030 PLOT 85,x+size,y
1040 PLOT 85,x+size,y+size
1050 ENDPROC
```

By including the procedure in a loop as shown below, we can randomly display coloured squares on the screen.



```
10 REM Program FILLSQUARE2
20 REM Version B1.1
30 REM Author Mike Williams
40 REM BEEBUG April 1984
50 REM Program subject to Copyright
60 :
```

We can use a similar procedure to draw rectangles, or any four sided filled shape, and indeed shapes of more than four sides.

ALTERNATIVE TECHNIQUE

There is a quite different technique that can be used to produce coloured rectangles. This uses the VDU24 command to define a graphics window on the screen. The required colour is then set as the background colour by using GCOL0,128+C where C is the colour in the range 0-15 (for Mode 2 at least). Simply executing the CLG command will then 'paint' the graphics window in the colour specified. The following short routine will draw the same yellow - coloured rectangle as before.

```
100 MODE 2
110 GCOL 0,129
120 VDU 24,200;100;800;500;
130 CLG
140 END
```

This technique is worth knowing, as it is quite short and quick, but it is more limited as only rectangles (or squares) may be drawn, and the rectangle must be wholly within the normal screen area, or the VDU24 command has no effect. You may also need to cancel the VDU24 command after use with VDU26 to avoid any lasting and unwanted effects.

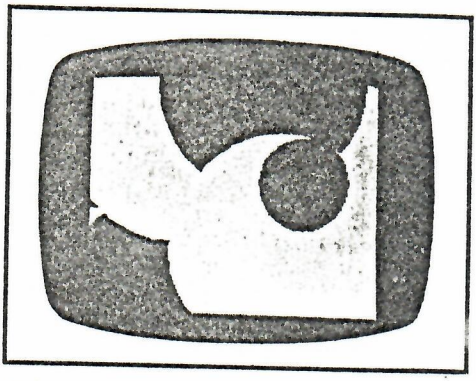
FILLED CIRCLES

We will now look at how we can use the triangle filling facility to generate filled circles. As we saw last month, a circle is just a polygon with a large number of sides. We can readily amend our circle program from last month to produce a filled circle, made up of many triangles, all touching at the centre.

```
100 MODE 2: GCOL 0,1
110 VDU29,640;512;
120 radius=200
130 MOVE radius,0
140 FOR angle=0 TO 2*PI STEP PI/16
150 x=radius*COS(angle)
160 y=radius*SIN(angle)
170 MOVE 0,0:PLOT 85,x,y
180 NEXT angle
190 END
```

Each time the program moves round the circumference from one point to the next, it fills in a triangle from those two points to the centre of the circle. The order of visiting the three points of the triangle is important in producing an efficient routine. This is again low resolution graphics (Mode 2) and it is worth experimenting with medium and high resolution (Modes 1 and 0) to see the effects produced. If you compare the program above with the one written last month for drawing an outline circle you should see many similarities.

We can easily turn our filled circle routine into a procedure, either by following the pattern of last month's fast circle drawing procedure, first setting up a table of sines and cosines to speed up the subsequent circle drawing, or by using the basic but slower routine used above.



This month, I have chosen to use the simpler method and the results are shown in the program below which displays circles of random size and colour, randomly positioned on the screen.

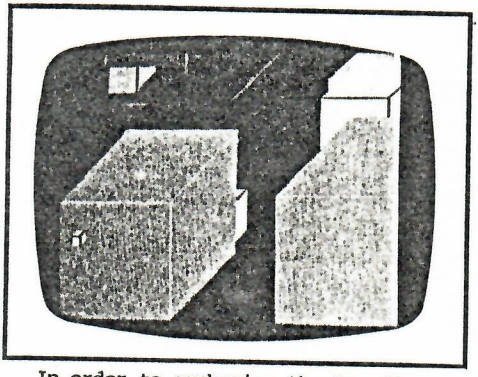
```
  10 REM Program FILLCIRCLE3
  20 REM Version E1.1 09/03/84
  30 REM Author Mike Williams
  40 REM ELBUG April 1984
  50 REM Program subject to Copyright
  60 :
 100 MODE 2
 110 ON ERROR GOTO 190
 120 REPEAT
 130 colour=RND(7):radius=RND(600)
 140 x=RND(1279):y=RND(1023)
 150 PROCfillcircle(colour,radius,x,y)
 160 UNTIL FALSE
 170 END
 180 :
 190 ON ERROR OFF:MODE 6
 200 REPORT:PRINT" at line ";ERL:END
 210 :
1000 DEF PROCfillcircle(colour,radius,
x,y)
1010 LOCAL angle,X,Y
1020 VDU29,x;y;:GCOL 0,colour
1030 MOVE radius,0
1040 FOR angle=0 TO 2*PI STEP PI/16
1050 X=radius*COS(angle)
1060 Y=radius*SIN(angle)
1070 MOVE 0,0:PLOT 85,X,Y
1080 NEXT angle
1090 ENDPROC
```

This last program also shows very clearly that, just as with DRAW and MOVE, no error occurs if the shape being drawn goes off the visible screen area.

3D SHAPES

The final program for this month shows how the three commands of MOVE, DRAW and PLOT can be combined together in a simple procedure that gives an appearance of depth to the screen display. Most of the work in this program is done by a procedure called PROCcube. The purpose of this procedure is to display a cube on the screen with a specified colour, size and position. This is really just an extension of our earlier filled square procedure.

A cube, as displayed on the screen, consists of a square (just as before) together with two paralellograms, one adjacent to the top edge and one adjacent to the right-hand edge of the square. The furthest visible corner of the cube is chosen to be 0.6 of the size of the cube in distance from these two edges. This is quite arbitrary, the value really depending on the effect you want to achieve. This is not a true perspective, as opposite sides are parallel, not converging to a vanishing point, but at least it keeps the maths quite simple.



In order to emphasize the 3D effect, the three edges of the cube separating the visible faces are drawn over the top of the coloured faces using a colour logically one more than that used for the cube as a whole.

As with the other programs, this one repeatedly calls the procedure to place

cubes randomly on the screen. Nevertheless the results are both colourful and effective.

```
 10 REM Program CUBES
 20 REM Version E1.0
 30 REM Author Mike Williams
 40 REM ELBUG April 1984
 50 REM Program subject to Copyright
 60 :
100 MODE 2
110 ON ERROR GOTO 190
120 REPEAT
130 x=RND(1024):y=RND(1024)
140 size=RND(500):colour=RND(7)
150 PROCcube(colour,size,x,y)
160 UNTIL FALSE
170 END
180 :
190 ON ERROR OFF:MODE 6
200 REPORT:PRINT" at line ";ERL:END
210 :
1000 DEF PROCcube(colour,size,x,y)
1010 GCOL 0,colour
1020 MOVE x,y:MOVE x,y+size
1030 PLOT 85,x+size,y:PLOT 85,x+size,y
+size
1040 PLOT 85,x+1.6*size,y+0.6*size
1050 PLOT 85,x+1.6*size,y+1.6*size
1060 MOVE x+size,y+size
1070 PLOT 85,x+0.6*size,y+1.6*size
1080 PLOT 85,x,y+size
1090 GCOL 0,(colour+1)MOD8
1100 DRAW x+size,y+size:DRAW x+size,y
1110 MOVE x+size,y+size
1120 DRAW x+1.6*size,y+1.6*size
1130 ENDPROC
```

Next month we will look at ways of filling more irregularly shaped areas on the screen, together with some of the other applications of the PLOT command.

HINTS HINTS HINTS HINTS HINT

CHANGING THE FLASH RATES

The rate at which the flashing colours flash, can be changed using *FX9,m and *FX10,s where m and s are described below.

*FX9,0 forces the first named colour to show continually.
*FX9,m sets the mark duration for the first colour, with m in 50ths of a second
*FX10,0 forces the second named colour to show continually.
*FX10,s sets the space duration for the second named colour, with s in 50ths of a second.
Initially the values of m and s are 25.

SPACE INVADER PROMPT

The VDU 23 call can be used very amusingly to re-define the prompt symbol '>' to be a space invader. Try the following:

```
VDU23,62,60,126,219,255,126,60,36,66 <return>
```

POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG

Dear Sir,

Due to the shortage of Electron software on the market, I thought it would be a good idea to publish a list in your magazine of all the BBC micro software which will run reasonably well on the Electron. The games which I have found to work well are:

DARE DEVIL DENNIS
by Vision Software.
GALACTIC COMMANDER by Micro Power.
MONSTERS by Acornsoft.
3D-BOMB ALLEY by Software Invasion.
VORTEX by Software invasion.

These games work, though any logos or title pages appear as gibberish. Finally I would like to congratulate you on your superb magazine. I find it interesting, useful and fun, and the games are always well worth the effort of typing them in.

M.Ryemill

Reply:

We hope other members will find this useful, though there are now versions of Dare Devil Dennis and Monsters available for the Electron, and other programs will soon follow.

SAVING AND LOADING SCREEN DISPLAYS

by David A. Fell

On the Introductory Cassette that comes with each Electron, there are two programs called ISLAND and PLANETS which each load a 'picture' from cassette to the screen, and display two apparently moving displays: one of an island, and one of a number of planets against a starry background. David Fell now explains how to load and save such screens to cassette, and presents a procedure to help you.

When the Electron displays a screen, it is reading data from an area of memory, called 'screen memory', and converting it into electrical signals that are fed to your TV (or monitor). Thus, it is the contents of the screen memory that dictate what will be shown on your display, and by altering this memory, we can alter what is displayed. With a cassette recorder attached to your Electron, you can easily save the contents of the screen memory, and load it back in again at a later date in such a way as to exactly reproduce the picture first obtained. We present here a procedure which you may append to any screen generating program in Basic, to allow screens to be saved to cassette. It is easy to use, and does not even require you to tell it what screen mode is in use - it works it all out for itself. To use this procedure, proceed as follows:

a) Type in the procedure, and save it to tape in the way described in the article 'Using Functions and Procedures' found elsewhere in this issue.

b) Load the program to which the procedure is to be appended, and check that the line numbers do not clash. If they do, you should renumber the main program first.

c) Exec in the procedure (in the manner described in the above mentioned article).

You should now have a copy of your program with the screen procedure appended. Next simply insert the following line into your main program at the point where you wish the screen to be saved:

PROCsavescreen

Whenever the program comes across this command, the screen saving procedure will be entered. This will produce a beep to tell you to start your recorder (in record mode), and another beep when the screen has been saved. Visual prompts obviously cannot be used, since they would interfere with the display to be saved.

There is one limitation to the saving of screens in this way - the screen must NOT have been scrolled after the last change of mode before the save is performed (unless you are using 'windows').

RE-LOADING THE SCREEN

The screen saved using the procedure presented here will be given the name SCREEN. To re-load it, change to the correct mode (the mode in which the screen was saved), and then type:

*OPT1 <return>
*LOAD SCREEN <return>

The * is vital, and SCREEN does not need to be enclosed in quotes. As it loads, you should see your picture build up from the top downwards.

NOTE: The command *OPT1 prevents any of the normal screen messages appearing while saving the screen image. You can return to the normal state by typing
*OPT <return>
or by just pressing Break.

```
1000 DEF PROCsavescreen
1010 *OPT1
1020 LOCAL A%,B%
1030 A%=135:A%=USR(&FFF4)AND&FF0000
1040 A%=A%DIV&10000
1050 IF A%<3 B%=&3000
1060 IF A%=3 B%=&4000
1070 IF A%=4 OR A%=5 B%=&5800
```

```
1080 IF A%=6 B%=&6000
1090 VDU7
1100 *FX138,0,13
1110 OSCLI("SAVE SCREEN "+STR$~B%+"
7FFF")
1120 *OPT
1130 ENDPROC
```

TECHNICAL NOTES

You can also save the screen by typing in the command directly (or include the command directly in your program) by means of the operating system *SAVE command. This is used in the form of:

*SAVE <filename> <start address> 7FFF

where 'filename' is the name of the file that you want the screen to be saved as, and 'start address' is a hexadecimal number representing the start address of the screen memory. The table below shows the start address of screen memory for the 7 screen modes of the Electron.

MODE	START ADDRESS
0	&3000
1	&3000
2	&3000
3	&4000
4	&5800
5	&5800
6	&6000

If you manage to create any nice screen images that you think might be of interest to other readers, just save them on a cassette, and send them in to us at the editorial address. We will publish the best of these in the magazine and will award as a prize, to the author of the most outstanding display, one of the new SIR ROM expansion boards for the Electron (see our review of this new product elsewhere in this issue). The closing date for your entries is 30th April, and the Editor's decision will be final.

HINTS HINTS HINTS HINTS HINTS

EASIER ENTRY OF LOWER CASE VARIABLE NAMES - K.Walker, J.Hewes and many others

When in normal upper case mode, if shift is pressed with a letter key, the letter comes out in lower case. This mechanism allows lower case text to be very easily entered as part of a program. This is particularly useful for entering variable and procedure names in lower case characters.

RANDOM EVENTS

If you want something in a program to happen at random you should use the RND(n) command. For example to make PROCEXPLODE happen on average once in every six tries use:

```
IF RND(6)=1 THEN PROCexplode
```

TWO KEY ENTRY FOR ENDPROC

The Electron allows a number of Basic commands to be entered just using 'Func' and a single key. For example, Func-Z and Func-X can be used to produce the keywords END and PROC respectively. However, if you combine both keys, the keyword ENDPROC is formed; i.e. press and hold down the Func key, and then press Z followed by X.

EASIER VIEWING OF TEXT - N.Haigney

To make your Electron have the same screen appearance as that obtained on the "COMPUTER PROGRAMME" on television (blue with black stripes), use

```
*KEY1 MODE 6:VDU 19,0,4,0,0,0|M <return>
```

Pressing function key one will select Mode 6 and turn the background to blue. This is also one of the functions of the KEYSET program included on the ELBUG introductory cassette.

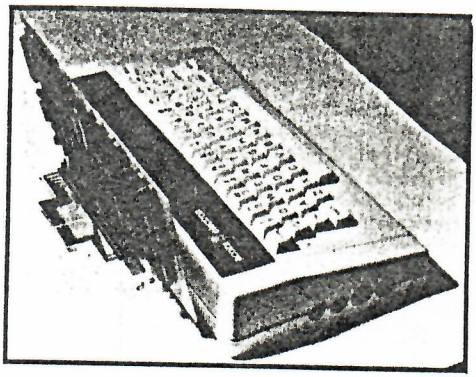
NEW ELECTRON ADD-ON FROM SIR COMPUTERS

Reviewed by David Fell and Philip Le Grand

We review here the new ROM expansion board for the Electron from Sir Computers, who seem to have won the race to produce the first Electron add-on.

WHAT IS A ROM BOARD?

You may well be wondering what a ROM board is and why you might need one. To explain, consider the case of the BBC micro. One of the features that makes this micro stand out from the competition is the ability to plug in special memory chips containing, for example, other programming languages and utility software. These memory devices are in two forms known as ROMs (Read Only Memories) or EPROMs (a reusable form of ROM). Their contents cannot be altered by writing to them, unlike the normal RAM (Random Access Memory) used for your own programs and data in your Electron. The BBC micro is capable of accessing up to sixteen different ROM devices, but the Electron does not include the necessary hardware for this task. This has prompted several hardware manufacturers to design suitable add-on units to rectify this situation. Sir Computers of Cardiff appear to have won the race to produce the first production-line product.



The Sir add-on ROM pack complete unit consists of two circuit boards enclosed in a case and a manual. The whole unit plugs in to the rear of the Electron and itself has a further identical connector allowing for even more expansion.

The unit is very large - the main circuit board used is slightly larger than the Electron's main circuit board - and is constructed to a very high standard. However, it is a pity that the high quality of the design did not continue on to the edge of this board. Our review board used several small edge connectors, which did not appear to be very strong, in order to connect the two boards that are used together.

We plugged a BBC micro Graphics ROM into the expansion board, and on connecting the unit to an Electron, switching on, and typing

*HELP <return>

the following message was displayed:

GRAPHICS EXTENSION 1.03

OS 1.00

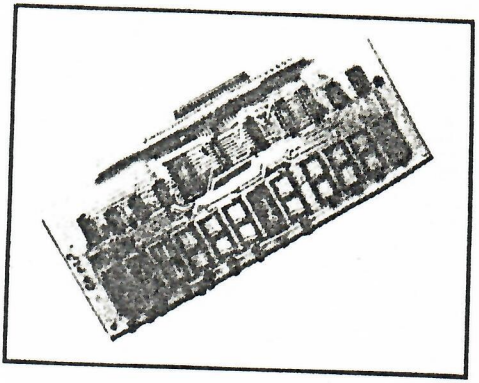
This proved that the board was working - at least to the extent of being able to recognise the Graphics ROM. In fact almost all the commands offered by this ROM worked well on the Electron.

The board was then filled to its capacity of twelve BBC micro ROMs - there being no available Electron ROMs at present. The Electron's power supply withstood the extra load imposed upon it by this. However, the majority of the software tested failed to operate correctly, if it worked at all. This is in no way due to Sir's add-on unit. In fact the only ROMs which operated correctly were VIEW and BCPL, both written by Acornsoft. The high failure rate was largely due to the extensive use of Mode 7 which is not available on the Electron. The Acornsoft ROMs are not mode dependent.

The board offers the user several configurations for ROM and RAM. All the options are link selectable, i.e. no soldering is required. The first four sockets can be configured to accept any combination of 2K, 4K, 8K, or 16K ROM or static RAM devices. The ability to mix devices allows ROM based software to be fully self contained with its own RAM workspace, without taking any of the precious user RAM, but any extra RAM added to the board in this way cannot be used to extend the main memory area of the Electron. The rest of the sockets will accept 8K or 16K ROM devices, but can be individually changed to take 2K or 4K devices.

Despite the apparently precarious connection of the ROM board to the Electron, which is somewhat reminiscent of the old ZX81 RAM pack, the computer failed to crash despite repeated attempts on our part to induce failure. We understand from Sir Computers that both boards, in production models, will be enclosed in a single case, the back of which will form a direct image of the rear of the Electron, allowing other peripherals to be added. Our review sample, a pre-production unit, was without a case.

It would not be fair to comment too much on the manual we received since it was a pre-release version, although the text and diagrams in it were clear and precise about the various options available to the user. The final version will, we understand, be a small booklet with a couple of extra example programs showing how to use the RAM options.



In conclusion, the expansion system performed well and allowed great flexibility in its use. The complete unit costs £47.15 (inclusive) which is not unreasonable compared to similar units for the BBC micro with less facilities. However, the fact remains that if you buy this board now, you will have virtually no software to use with it - though we will be producing two ROMs for the Electron shortly - a machine code monitor (EXMON) and a Basic programmers aid (TOOLKIT). It is also likely that other software in ROM will become available for the Electron now that there is a means of using it.

The board can be purchased from:
Sir Computers Ltd., 91 Whitchurch Road, Cardiff, CF4 3JP
Tel:(0222)621813

HINTS HINTS HINTS HINTS HINTS

TRANSPARENT KEY DEFINITIONS - T.G.Ward

It is possible to change the function key definitions without corrupting a program currently in memory. This is achieved by previously having *SAVEd the memory contents of the key definition buffer (located from &B00 to &BFF). Then when you want to change the definitions, just *LOAD the required set of definitions into memory. The whole process of loading into memory will be transparent to any Basic program existing in memory. For example, initialise your function keys in the usual way, and then save the definitions as follows:

*SAVE "DEF1" B00 BFF <return>

Where DEF1 is the file name. Now, whenever this set of functions are required, type in the following line:

*LOAD "DEF1" <return>

USING PROCEDURES AND FUNCTIONS

by Rob Pickering and Philip Le Grand

If you have looked in any detail at many of the programs published in ELBUG you will have seen that the majority make extensive use of procedures. These, together with functions, form a very useful and powerful feature of BBC Basic (used on the Electron). Rob Pickering and Philip Le Grand explain clearly just what functions and procedures are and tell you how to set up your own function and procedure library.

WHAT IS A PROCEDURE?

Although it is perfectly feasible to write programs for the Electron without any knowledge of procedures, there is no doubt that their correct usage leads to a far superior program.

Think of a short part of a whole program, which carries out one minor task within the program. A few lines which, for example, wait for the user to press a key before continuing. Assume that this minor task will be performed several times within the program, which is fairly typical. We could enter the same lines at each stage of the program where it is required, but this is tedious and takes up a lot of memory. What we do is make the relevant group of lines into a collective unit called a procedure. So you can think of a procedure as being a section of program which performs a given task as many times as required. It is similar in some ways to a subroutine (i.e. using GOSUB...RETURN) but has certain advantages over the subroutine.

Each procedure is given a name to distinguish it from other procedures. A procedure name is very similar to a variable name, but with two real differences:

(a) they must always start with the keyword "PROC" in order to distinguish them from variable names,

(b) they may not end with a "$" or "%" symbol.

Choosing an appropriate name for a procedure is very important, not only to make the program understandable to other people, but also to help yourself to remember what all the procedures do. Since the purpose of our example procedure is to wait for the user to press a key, it seems sensible to call it "PROCwait". Note also that using lower case letters for the name helps to make it clearer to read. Here then is our example procedure in full:

```
1000 DEF PROCwait
1010 *FX15,1
1020 key=GET
1030 ENDPROC
```

As you can see this is only a simple short example and as such it is not typical of the usual length of procedures. You will notice that line 1000 starts with the letters "DEF", short for DEFINITION. It simply informs the computer that this is the start of a procedure that we are 'defining'. Once defined, we simply state its unique name whenever it is required. For instance, here is an example of how we might use PROCwait within a program.

```
100 PRINT"Press any key to continue.."
110 PROCwait
120 PRINT"Thankyou."
130 END
```

Notice that we do not need to know at what program line the procedure starts in order to use it. The BBC Basic holds a list of where the procedures are in the program, each location being stored when the procedure is defined. This means that when you use a procedure, Basic does not have to search through the whole program in order to find the correct procedure, hence providing fast operation. Note also that procedures are terminated with an ENDPROC, passing execution back to the instruction following the one which called the procedure. Procedures can also be called in 'immediate mode' - that is to

say, directly from the keyboard. Thus if you type PROCwait <return>, the procedure will also be executed. This allows you to debug a program procedure by procedure, rather than trying to test a whole program at once.

PARAMETERS

You will inevitably have heard of parameters, but may not know what they are, or what their purpose is. Parameters form a handy way of passing values to a procedure. For example, we can alter the above procedure to cause a delay of a set amount, by passing the delay length as a parameter to the procedure rather than waiting for ever if a key is not pressed, so a sensible name would be "PROCdelay", as given below:

```
100 DEF PROCdelay(secs)
110 TIME=0
120 REPEAT
130 REM Do nothing
140 UNTIL TIME>100*secs
150 ENDPROC
```

A parameter is simply a value which will be used by the procedure in some way in the performance of its task. We use the expression "parameter passing" for the way in which values are passed from the main program into the procedure. Here, the procedure definition includes a parameter, (a variable named "secs") which will determine the number of seconds delay, and applies to a variable which will exist in the locality of the procedure only.

LOCAL AND GLOBAL VARIABLES

Something which exists only inside a procedure definition is said to be "LOCAL" to the procedure. A reference to something outside the procedure is said to be a "GLOBAL" reference. When a variable name is given as a parameter in the procedure definition it does not then use a variable of the same name from the main program. What happens is that when we use the procedure we must give a parameter there too. The value given in the call to the procedure is automatically copied into the procedure's own local variable. The type of parameters that can be passed to a procedure include real, integer and string variables, as well as actual values, but remember that the type of parameter MUST match the type of variable in the definition. Note though, arrays cannot normally be passed to procedures. An example will help to make this clearer.

```
 100 value1=3.526
 110 value2%=55:y%=10
 120 value3$="HELLO"
 130 PROCoutputvalues(value1,value2%,va
lue3$,101)
 140 PRINT y%
 150 END
1000 DEF PROCoutputvalues(x,x%,x$,y%)
1010 PRINT''x,x%,x$
1020 PRINT y%
1030 ENDPROC
```

Enter and run the complete program above, which demonstrates that the variables passed to the procedure are actually different from the variables in the rest of the program. In the main program "y%" is simply set to the value 10, but the fact that a variable of the same name within the procedure has a different value (namely 101) demonstrates that they are separate, and we can print out the original value of y% when we return. Also notice that the last parameter passed is a value and not a variable.

Local variables are not limited to those which we use as parameters. In fact we can make all the variables that are used within a procedure local to itself. To achieve this we are provided with the Basic command "LOCAL" which is followed by a list of all the local variables. Provided that any variables used within a procedure are declared as LOCAL, you never have to worry whether the same variable names are being used elsewhere, either in the main program or in another procedure. This means that you can take a procedure from someone else and need only know what task it performs, its name, and what parameters it requires to use it in your own program.

STRUCTURED PROGRAMS

One of the great advantages of using procedures is that their very use leads naturally to structured programs. As an example of a well structured program, see program 1 below (we have not included line numbers as this is just a

general outline of a program). It has two main sections. The first is called the MAIN section and contains generally very 'readable' code, calling upon each procedure to carry out its task when it is needed. The second section is merely a collection of all the procedure and function definitions. See if you can tell what it does without having any idea of the actual programming involved in the definition.

```
REM Program 1
PROCtitles
PROCinitialise
REPEAT
PROCoptionmenu
choice$=FNoption
IF choice$="C" PROCcreatefile
IF choice$="E" PROCeditfile
IF choice$="U" PROCupdatefile
UNTIL choice$="STOP"
PROCtidyup
END
REM *** DEFINE ALL PROCEDURES
DEF PROCoptionmenu
  ...
ENDPROC
DEF PROCcreatefile
  ...
ENDPROC
DEF PROCeditfile
  ...
ENDPROC
DEF PROCupdatefile
  ...
ENDPROC
DEF PROCtidyup
  ...
ENDPROC
DEF FNoption
  ...
=...
```

The 'END' statement dividing the two sections of program is essential here to avoid the program re-entering the procedure definitions after the main body of the program has been completed.

FUNCTIONS

You may also have noticed in the above program the line with 'choice$=FNoption' in it. This is another device used in structured programs. It is called a function. Functions are very similar to procedures except that they can return one value back to the main program. The value returned may be an integer, a real number or a string. The function is also called differently to a procedure, as in the example, by being used as would any value in the program. For example

```
ans$=FNoption
```

or PRINT FNoption

Functions are defined in a similar manner to procedures using the keyword DEF. The function FNoption could be defined as follows:

```
DEF FNoption
PRINT"C = creates a new file"
PRINT"E = edit a file"
PRINT"U = update a file"
PRINT"STOP = exit program"
PRINT'"Enter your choice "
INPUT entry$
=entry$
```

The result is always placed at the end of the function, after the '=' character, to terminate it. Similarly as many parameters as you like can be passed to a function, although only one is returned. It is possible to write complete programs using procedures and functions without resorting to GOTOs or GOSUBs of any kind.

USING FUNCTIONS AND PROCEDURES

Because of their versatility, many functions and procedures that you write can be used time and time again in your own programs. This is where a function/procedure library can be useful.

The idea of such a library is that it should contain all your most useful functions and procedures. When you are writing a new program, you will often find you can speed up program development by using some of the existing procedures and functions from your library. It is very easy to set up your own library using the *EXEC and *SPOOL commands.

When you have a function or procedure that you want to save, do make sure it is fully tested. After all, there is no value in having a library of failures! It is also a good idea to number the procedures with high line numbers, say from 10000 onwards. Then select a suitable place on your library cassette to save the module, and type in the following:

*SPOOL example <return>

where 'example' is the name of the module (no quotes are required). Switch your recorder to record when prompted by the computer, press Return and type LIST <return>. When the listing finishes, type *SPOOL <return>, and turn your cassette recorder off. Now every time you require that module in another program, locate its position on the tape, renumber your program so that the last line is less than 10000, and type:

*EXEC example <return>

Press Play on your recorder, and the module will be automatically merged on to the end of the program. This can be repeated for as many modules as you require. It is important that the line numbers of the program and the procedure being merged are quite separate. If necessary, renumber your program before merging in the next section.

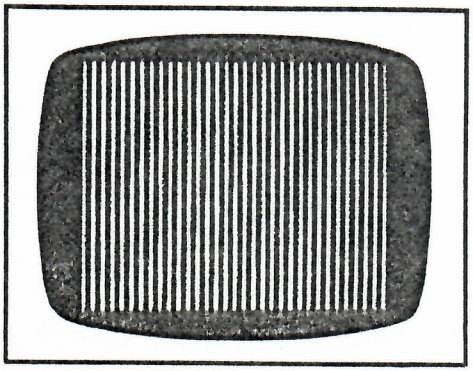
Some examples of functions and procedures that people have already set up into their libraries include fast circle plotting procedures, delay procedures, functions to read data input at the keyboard, procedures to plot double height characters, date validation procedures, and sorting procedures. We will be supplying some ready-made procedures in forthcoming issues of ELBUG. This issue, for example, contains a procedure for saving graphics screens.

FABRIC PATTERNS

by D. J. Allom

The following very short program creates a variable technicolour display, simulating woven patterns seen in fabrics. The display is built up from left to right with successive passes using randomly chosen colours. The program may be halted temporarily at the end of a complete pass by pressing the space bar once. When it is pressed again, the program continues its progress, changing the pattern yet again. Re-running the program should produce new patterns. You may wish to make it perform this automatically to produce a non-stop display. Just add the following:

```
200 GOTO 10
```



```
 10 REM Program FABRIC
 20 REM Author  D.J.Allom
 30 REM Version E1.1
 40 REM ELBUG   April 1984
 50 REM Program subject to copyright
 60:
100 ON ERROR MODE 6:END
110 MODE2:VDU 23,1,0;0;0;0;
120 REPEAT
130 GCOL3,RND(7):A%=RND(4)*4
140 FORB%=256 TO 1024 STEPRND(4)*8
150 FORC%=224 TO 800STEPA%
160 PLOT69,B%,C%
170 NEXT,
180 IF INKEY(0)=32 THEN OSCLI("FX15,1
"):REPEAT UNTIL GET=32
190 UNTIL FALSE
200 END
```

DOMINOES

by C. C. Radcliffe

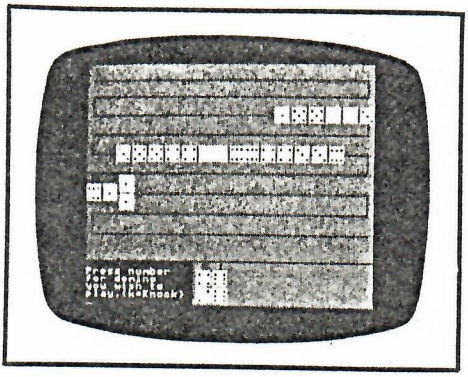
The program DOMINO provides an excellent computer version of the popular table game of dominoes for you to play against your Electron. The graphics used in this program are a real delight and show just what can be achieved using this micro.

This is a game of dominoes in which you pit your wits against the ice cool steely resolve of your Electron (or Elk as it likes to be known). The object of dominoes is very simple - you must try and lay all your dominoes down on the 'table' before the computer.

You and the computer each select a domino at the start of the game to see who goes first. You will then see your initial set of seven dominoes displayed at the bottom right of the screen. When it is your turn to go, choose which domino you want to play by pressing the appropriate number key, and then press either 1 or 2 to indicate at which end you wish to play. This may often be obvious to you, but not to the computer which cannot 'see' your hand. If you are unable to play a domino from your hand, when it is your turn, then you must 'knock' by pressing 'K', and you will be given another domino from the unused pile. The computer's dominoes are not displayed on the screen; that would be cheating, but it plays to exactly the same rules and will also 'knock' when unable to go.

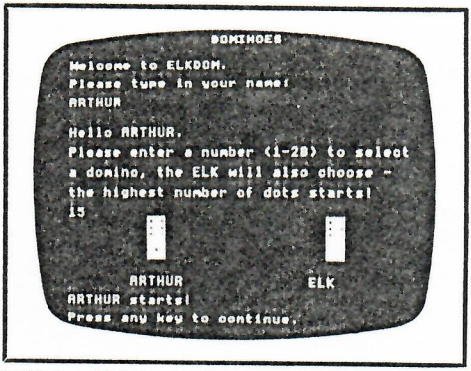
Note that because of the limitations of the screen display, you cannot have more than 9 dominoes in your hand at any one time. At the end of the game, the overall score is displayed on the screen, together with any dominoes left over in either hand.

This is a very nicely presented game using Mode 1 graphics which is really pleasing to play.



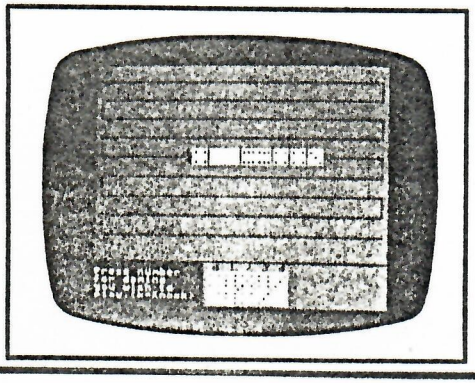
```
  10 REM PROGRAM DOMINOES
  20 REM AUTHOR  C.C.Radcliffe
  30 REM VERSION E1.0
  40 REM ELBUG   APRIL 1984
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 ON ERROR GOTO 430
 110 :
 120 CLEAR
 130 *FX11,0
 140 MODE1
 150 VDU23,1,0;0;0;0;
 160 PROCinit
 170 PROCshuffle
 180 PROCstart
 190 REPEAT
 200 WT=FALSE
 210 PROCshuffle:S%=0
 220 GCOL0,129:CLG
 230 VDU28,0,31,14,26,12,18,0,0,25,4,4
80;188;25,5,1279;188;
 240 FOR I%=1 TO 9
 250 MOVE29,1023-I%*80:PLOT25,1221,0
 260 IF (I%MOD2 AND I%<9) PLOT25,0,-80
 ELSE IF I%<9 PLOT0,-1221,0:PLOT25,0,-
80
 270 NEXT
 280 FORI%=0TO1
 290 PROCpick(I%,7):end%(I%)=-1:end1%(
I%)=3
 300 NEXT
 310 WT=TRUE
 320 REPEAT
 330 IFELK%PROCELK ELSEPROCdisphand(1,
0):PROCplayer
 340 UNTILtot%(0)=0 ORtot%(1)=0 OR(bk%
ANDpk%ANDS%>27)
 350 I%=0:W%=0:REPEAT
 360 IFtot%(I%)=0 PROCwin(I%):W%=1
 370 I%=I%+1
 380 UNTIL W% OR I%=2
```

```
  390 IFW%=0 W%=FNwin:PROCwin(W%)
  400 UNTILFALSE
  410 END
  420 :
  430 ON ERROR OFF:MODE 6
  440 *FX12
  450 IF ERR<>17 REPORT:PRINT" at line
";ERL
  460 END
  470 :
 1000 DEFPROCinit
 1010 DIMdom$(27),hand$(1,9),tot%(1),en
d%(1),end1%(1),endx%(1),win%(2),th%(1)
 1020 A%=RND(TIME)
 1030 FORI%=0TO1
 1040 tot%(I%)=0
 1050 FORK%=0TO9
```



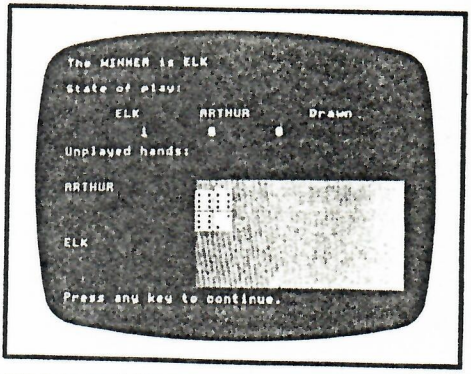

```
 1060 hand$(I%,K%)="  "
 1070 NEXT:NEXT
 1080 FOR I%=0 TO2:win%(I%)=0:NEXT
 1090 VDU23,255,0;0;32,112,32,0
 1100 VDU19,1,2,0,0,0:VDU19,2,1,0,0,0
 1110 bk%=0:pk%=0
 1120 ENDPROC
 1130 :
 1140 DEFPROCshuffle
 1150 RESTORE1280
 1160 LOCALN%,I%
 1170 FORN%=0TO27
 1180 dom$(N%)="  "
 1190 NEXT
 1200 FORN%=0TO27
 1210 REPEAT
 1220 I%=RND(28)-1
 1230 UNTILdom$(I%)="  "
 1240 READdom$(I%)
 1250 NEXT
 1260 ENDPROC
 1270 :
 1280 DATA00,01,02,03,04,05,06,11,12,13
,14,15,16,22,23,24,25,26,33,34,35,36,44
,45,46,55,56,66
 1290 :
 1300 DEFPROCstart
 1310 LOCALB%,bd%,P%,N%,pd%,s$,t$
 1320 CLS
 1330 PRINTTAB(16,1)"DOMINOES"
 1340 PRINT''"Welcome to ELKDOM."''"Ple
ase type in your name:"
 1350 REPEAT:INPUTTAB(0,8)n$:L%=LEN(n$)
:IFL%>10PRINTTAB(0,8)SPC(L%)
 1360 UNTILL%<11
 1370 PRINT''"Hello ";n$;"."
 1380 PRINT"Please enter a number (1-28
) to select"''"a domino, the ELK will a
lso choose -"''"the highest number of d
ots starts!"
 1390 REPEAT:INPUTTAB(0,19)P%:P%=P%-1:U
NTILP%<28ANDP%>=0
 1400 pd%=FNtotdot(P%)
 1410 REPEAT
 1420 B%=RND(28)-1
 1430 bd%=FNtotdot(B%):s$="ELK"
 1440 UNTILC%<>B%ANDpd%<>bd%
 1450 PROCdomino(320,320,TRUE,FN1dot(do
m$(P%)),FNrdot(dom$(P%))):PROCdomino(96
0,320,TRUE,FN1dot(dom$(B%)),FNrdot(dom$
(B%)))
 1460 PRINTTAB(0,26)SPC(10-LEN(n$)DIV2)
;n$;SPC(17-LEN(n$)DIV2);s$
 1470 IFbd%>pd%ELK%=TRUEELSEELK%=FALSE:
s$=n$
 1480 PRINT's$;" starts!"''"Press any k
ey to continue."
 1490 P%=GET
 1500 ENDPROC
 1510 :
 1520 DEFFNtotdot(N%)=FN1dot(dom$(N%))+
FNrdot(dom$(N%))
 1530 :
 1540 DEFFN1dot(D$)=VAL(LEFT$(D$,1))
 1550 :
 1560 DEFFNrdot(D$)=VAL(RIGHT$(D$,1))
 1570 :
 1580 DEFPROCdomino(X%,Y%,O%,L%,R%)
 1590 LOCALN%,D%
 1600 GCOL0,2:IFO%MOVEX%-30,Y%-6:PLOT0,
0,12:PLOT81,60,-12:PLOT81,0,12ELSEMOVEX
%-6,Y%-30:PLOT0,12,0:PLOT81,-12,60:PLOT
81,12,0
 1610 FORN%=-1TO1STEP2
 1620 GCOL0,3:IFO%MOVEX%+N%*30,Y%+N%*66
ELSEMOVEX%+N%*66,Y%+N%*30
 1630 PLOT0,-60*N%,0:PLOT81,60*N%,-60*N
%:PLOT81,-60*N%,0
 1640 NEXT
 1650 O%=ABS(O%)
 1660 FORN%=-1TO1STEP2
 1670 H%=O%*(X%-8)-(O%-1)*((X%-8)+N%*36
 1680 V%=O%*((Y%+18)+N%*36)-(O%-1)*(Y%+
18)
 1690 IFN%=-1D%=L%ELSED%=R%
 1700 VDU5:PROCdot(H%,V%,D%)
 1710 NEXT
```

```
 1720 VDU4
 1730 ENDPROC
 1740 :
 1750 DEFPROCdot(X%,Y%,T%)
 1760 LOCALN%,M%,D%,A%,H%,V%
 1770 M%=T%MOD2:D%=T%DIV2
 1780 GCOL4,0:MOVEX%,Y%
 1790 IFM%PRINTCHR$255;
 1800 IFD%=0ENDPROC
 1810 IFO%RESTORE1940ELSERESTORE1950
 1820 N%=0
 1830 REPEAT
 1840 READA%:N%=N%+1
 1850 UNTILN%=D%
 1860 RESTORE1960
 1870 FORN%=1TO8
 1880 READH%,V%,Q%
 1890 MOVEX%,Y%:PLOT0,H%,V%
 1900 IFQ%ANDA%PRINTCHR$255
 1910 NEXT
 1920 ENDPROC
 1930 :
 1940 DATA&44,&55,&DD
 1950 DATA &11,&55,&77
 1960 DATA -20,20,1,0,20,2,20,20,4,20,0
,8,20,-20,&10,0,-20,&20,-20,-20,&40,-20
,0,&80
 1970 :
 1980 DEFPROCpick(P%,D%)
 1990 LOCALN%
 2000 FORN%=1TOD%
 2010 IFS%>27D%=N%ELSEhand$(P%,tot%(P%)
)=dom$(S%):S%=S%+1:PROCsound
 2020 NEXT
 2030 ENDPROC
 2040 :
 2050 DEFPROCdisphand(P%,T%)
 2060 VDU24,500;0+T%*184;1279;184+T%*18
4;16
 2070 LOCALN%,X%
 2080 IF tot%(P%)=0 VDU26:ENDPROC
 2090 FORN%=0 TOtot%(P%)-1
 2100 X%=532+(N%)*72
```

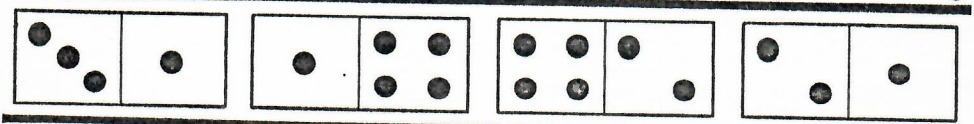


```
 2110 PROCdomino(X%,80+T%*184,TRUE,FN1d
ot(hand$(P%,N%)),FNrdot(hand$(P%,N%)))
 2120 MOVEX%-10,180+T%*184
 2130 VDU5
 2140 IF T%=0 PRINT;N%
 2150 NEXT:IF T%=0 VDU4,24,0;196;1279;1
023; ELSE VDU4,26
 2160 ENDPROC
 2170 :
 2180 DEFPROCELK
 2190 LOCALE%,N%,D%,O%,L%,R%,P%:ELK%=FA
LSE:bk%=0
 2200 IFend%(0)=-1X%=640:Y%=623:D%=FN1s
t:L%=FN1dot(hand$(0,D%)):R%=FNrdot(hand
$(0,D%)):end%(0)=L%:end%(1)=R%:O%=(L%=R
%):P%=1:FORE%=0 TO1:endx%(E%)=640+FNcen
tre:PROCendno(E%):NEXTELSEPROCplay:PROC
endno(E%)
 2210 IFP%PROCdomino(X%,Y%,O%,L%,R%):PR
OCsort(0,D%)ELSEPROCpick(0,1):bk%=1
 2220 ENDPROC
 2230 :
 2240 DEFFNvert(D$)
 2250 IFFN1dot(D$)=FNrdot(D$)=TRUEELSE=
FALSE
 2260 :
 2270 DEFPROCplayer
 2280 LOCALE%,B%,F%,N%,P%,D%,X%,Y%,O%,I
%,J%,L%,R%:ELK%=TRUE:pk%=0
 2290 PRINT'"Press number""for domino"
'"you wish to""play.(k=Knock)"
 2300 REPEAT:D%=GET-48:UNTIL(D%>=0 ANDD
%<tot%(1))ORD%=59 ORD%=27:IFD%>tot%(1)P
ROCpick(1,1):pk%=1:ENDPROC
 2310 I%=FN1dot(hand$(1,D%)):J%=FNrdot(
hand$(1,D%)):O%=FNvert(hand$(1,D%))
 2320 IFend%(0)=-1X%=640:Y%=623:L%=I%:R
%=J%:end%(0)=L%:end%(1)=R%:P%=TRUE:FORE
%=0 TO1:endx%(E%)=640+FNcentre:PROCendn
o(E%):NEXT:GOTO2370
 2330 IFend%(0)<>-1CLS:PRINT'"Press num
ber""for end""you wish to""play.":RE
PEAT:E%=GET-49:UNTILE%=0ORE%=1
 2340 IFI%=end%(E%)B%=0:F%=1
 2350 IFJ%=end%(E%)B%=1:F%=1
 2360 IFF%PROCendno(E%):PROCarrange(B%)
:PROCsetup(E%):PROCendno(E%)
 2370 IFP%PROCdomino(X%,Y%,O%,L%,R%):PR
OCsort(1,D%)ELSEPROCpick(1,1):pk%=1
 2380 IF WT SOUND0,-15,100,6
 2390 IF WT G=INKEY(100)
 2400 ENDPROC
 2410 :
 2420 DEFPROCplay
 2430 LOCALI%,i%,J%,j,K%,B%,T%,M%:E%=-1
 2440 FORM%=0TOtot%(0)-1
 2450 i%=FN1dot(hand$(0,M%)):j%=FNrdot(
hand$(0,M%))
 2460 FORK%=0TO1
 2470 IFi%=end%(K%)ANDT%<=i%+j%PROCsave
:B%=0
```

```
 2480 IFj%=end%(K%)ANDT%<=i%+j%PROCsave
:B%=1
 2490 NEXT:NEXT
 2500 O%=FNvert(hand$(0,D%))
 2510 IFE%>-1 PROCendno(E%):PROCarrange
(B%):PROCsetup(E%)
 2520 ENDPROC
 2530 :
 2540 DEFPROCsave:D%=M%:T%=i%+j%:I%=i%:
J%=j%:E%=K%:ENDPROC
 2550 :
 2560 DEFPROCsetup(E%)
 2570 LOCALC%,W%
 2580 P%=1:C%=FNcentre:X%=endx%(E%)+C%:
W%=endx%(E%)+2*C%+8*SGN(C%)
 2590 IFW%<70RW%>1271end1%(E%)=end1%(E%
)+SGN(C%):X%=C%-1279*(E%=0):endx%(E%)=W
%-endx%(E%)-1279*(E%=0)ELSEendx%(E%)=W%
 2600 Y%=1103-end1%(E%)*160
 2610 ENDPROC
 2620 :
 2630 DEFFNcentre=-((O%=0)*70+(O%=TRUE)
*34)*((E%=0)-(E%=1))
 2640 :
 2650 DEFPROCsort(P%,D%)
 2660 LOCALN%,T%
 2670 IFtot%(P%)<10T%=tot%(P%)-1ELSET%=
8:hand$(P%,9)="  "
 2680 FORN%=D%TOT%
 2690 hand$(P%,N%)=hand$(P%,N%+1)
 2700 NEXT
 2710 tot%(P%)=tot%(P%)-1
 2720 ENDPROC
 2730 :
 2740 DEFPROCarrange(F%)
 2750 IFF%=0end%(E%)=J%ELSEend%(E%)=I%
 2760 IFF%=E%R%=I%:L%=J%ELSEL%=I%:R%=J%
 2770 ENDPROC
 2780 :
 2790 DEFFNlst
 2800 LOCALN%,T%,H%,I%
 2810 FORN%=1TO7
 2820 T%=FNtotdot(N%-1):IFT%>H%H%=T%:I%
=N%-1
 2830 NEXT:=I%
 2840 :
 2850 DEF PROCendno(E%)
 2860 IFE%=-1 ENDPROC ELSE VDU5:GCOL4,0
:MOVEendx%(E%)+30*(E%=0),1115-end1%(E%)
*160:PRINT;E%+1:VDU4
 2870 ENDPROC
 2880 :
 2890 DEF FNwin
 2900 LOCAL N%,I%
 2910 FOR N%=0 TO 1
 2920 th%(N%)=0
 2930 FOR I%=0 TO tot%(N%)-1
 2940 th%(N%)=th%(N%)+FNldot(hand$(N%,I
%))+FNrdot(hand$(N%,I%))
 2950 NEXT:NEXT
 2960 IF th%(0)>th%(1) =1
 2970 IF th%(1)>th%(0) =0 ELSE =2
 2980 :
 2990 DEF PROCwin(W%)
 3000 VDU26,12
 3010 LOCALN%
 3020 IF W%=0:ELK%=TRUE:W$="ELK"
 3030 IF W%=1:ELK%=FALSE:W$=n$
 3040 IF W%<2 PRINT'"The WINNER is ";W$
 ELSEPRINT'"Game Drawn!"
 3050 PRINT''"State of play:"
 3060 win%(W%)=win%(W%)+1
 3070 PRINT''SPC(6);"ELK";SPC(10-LEN(n$
)DIV2+LEN(n$)MOD2);n$;SPC(10-LEN(n$)DIV
2);"Drawn"''SPC(2);
 3080 FOR N%=0 TO 2
 3090 PRINT;SPC(7);win%(N%);
 3100 NEXT
```




```
 3110 PRINT''"Unplayed hands: "'''''n$''
'''''"ELK"''''
 3120 PRINT''"Press any key to continue
."
 3130 FOR N%=1 TO2:PROCdisphand(N%-1,N%
):tot%(N%-1)=0:NEXT
 3140 k=GET
 3150 ENDPROC
 3160 :
 3170 DEFPROCsound
 3180 IF WT SOUND1,-15,100,4:SOUND1,-15
,10,6
 3190 IF WT G=INKEY(100)
 3200 IF tot%(P%)<9 tot%(P%)=tot%(P%)+1
 3210 ENDPROC
```



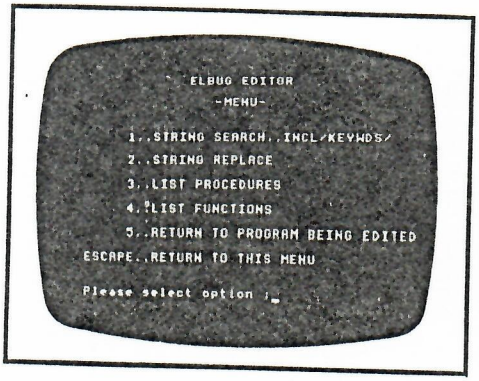
UTILITY EDITOR

by Adrian Calcraft, Paul Otto and T. Burgess

This excellent Utility Editor should prove an invaluable aid to Basic programmers. It allows you to search a program for Basic keywords, strings, procedures and functions, and to make global alterations.

When developing a program much time can be wasted scanning the listing to locate a specific procedure, function or character string that needs to be changed. The program listed with this article is a mini Editor, which can be loaded into the computer along with your development program. It is menu-driven, with the following options:

1. List all occurrences of a specific string, including Basic Keywords.
2. Change all occurrences of a specific string to any new string that is equal in length or shorter than the original string, thus allowing you globally to change variable names.
3. List the starting line number and name of each procedure used.
4. List the starting line number and name of each function used.
5. Return to the Basic program being edited.
6. Return to the menu.



TO SET UP THE EDITOR

Type in the Editor as listed, and check your version against the original listing. Then save a copy to tape under the file name "EDITOR".

TO USE THE EDITOR

1. Put the program that you wish to edit into memory using:
   LOAD "" <return>
2. Type:
   PAGE=&5000 <return>
   This resets the pointers in the Electron to allow the Editor to be resident in your machine at the same time as the program to be edited.
3. Load and Run the Editor using:
   CHAIN "EDITOR"
4. Use options 1 to 4 on the menu, as detailed below. Pressing Escape at any time will return you to the menu.
5. To exit from the Editor and return to the development program, use option 5.
6. To return to the Editor at any time (once you have exited), press function key 0 (ie press FUNC and 0 simultaneously.

ABOUT THE EDITOR

When you run the Editor you will be presented with a menu offering the 5 options. To choose an option simply select the number indicated. All other entries are locked out. If you enter 1 or 2 you will be prompted for a search string. Simply key in your string and then press Return. Option 2 will also prompt you for a replace string which you enter in the same way.

As already mentioned, option 1 will allow you to search for strings including a keyword. To tell the computer that the relevant part of the string is a keyword, it must be enclosed in 2 slashes (/). eg:

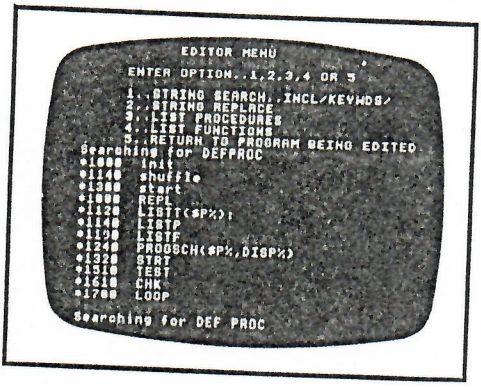
```
To search for PRINT "HELLO"
 enter /PRINT/ "HELLO"
To search for REPEAT
 enter /REPEAT/
To search for A%=A%+1:NEXTB%
```

```
enter A%=A%+1:/NEXT/B%
```

If the program cannot find the keyword specified it will print an error message, and continue the search with the string exactly as you entered it.

There are several points to note here:

1. The search will be made for the EXACT string which you supply. If the program contains: A% =A%+1 (ie with a space after the first %) and you search for A%=A%+1, it will not find it.
2. The Editor will only allow ONE keyword in any given string.
3. The string replace option may NOT be used with keywords.



If you choose options 3 or 4 no prompts are required, and the search for procedures or functions is initiated immediately. Having displayed the relevant line numbers (if there are any to display) the Editor will prompt you to press Return, upon which the menu will be redisplayed, ready for your next command. To return to your development program select option 5.

GENERAL NOTES

1. All options put the display in "paged mode". This prevents the located line numbers from being displayed, only to scroll off the top of the screen. If a full screen is displayed, pressing "Shift" will display the next screenful.
2. If you attempt to RUN the development program, it may well cause the Editor to be overwritten. If you need it again afterwards, you should load it in again as described above.

IF YOU HAVE PROBLEMS

If upon running the Editor you get an error message indicating that the computer has run out of room (eg. DIM space), it probably means that your version of the Editor program has become much longer than the original. If you wish to extend the Editor by adding further routines, you will need to load it in at a lower address - eg &4C00 in place of &5000. If you have to do this, don't forget to alter line 110 to reflect the new start address.

ERROR MESSAGES

"Error, try again". This is only issued by option 2 and indicates either that you have attempted to replace a string by a longer string, which is not allowed, or that the Editor cannot find the program to edit.

"Invalid Keyword". This is displayed by option 1 and indicates that you have attempted to search for a Basic Keyword which has not been found in the table e.g. /PRNT/. If this is specified as a search argument in option 1, the error message will be given. The search will continue, however, for the actual string /PRNT/.

ABOUT THE KEYWORD CONVERSION

The keyword table replacement routine will only be able to locate the keywords exactly as they are stored in the ROM memory. You will find it useful to know that the token for INSTR assumes the first bracket, and so should be searched for as /INSTR(/. This also applies for LEFT$, which is searched for as /LEFT$(/, MID$ which is /MID$(/, POINT which is /POINT(/, RIGHT$ which is /RIGHT$(/, STRING$ which is /STRING$(/ and TAB which is /TAB(/.

The following keywords are stored twice in the table, and only the first entry is accessed by the

conversion program. These are..PAGE, PTR, TIME, LOMEM, HIMEM.

If a search is requested for a string containing a keyword and nothing else, the program will be searching for a single byte (the token). It is possible that this byte may occur in a different context within the program, eg as part of a line number. If this is the case, the program will erroneously report a line number. This happens in practice very rarely, and only on searches for strings containing just a keyword and nothing else.

Searching for GOTO followed by a line number, eg /GOTO/200,will not work. This is because the line number after a GOTO, is stored by the computer in coded form.

TECHNICAL INFORMATION

BASIC KEYWORDS

As you may have heard, when the Electron stores a program in its memory, it converts all Basic keywords (such as IF, PRINT, SOUND, GOTO ...etc) into tokens. What this means is that rather than store all these, sometimes long, and frequently repeated words actually as they are spelt, it simply stores each of them as a 1 byte code (token).

This leads to much more economical use of storage space. However, it does result in a problem for string search programs; quite simply, searching for say, "PRINT" is a waste of time, as it has been stored in memory as 241, rather than as a sequence representing the letters "P-R-I-N-T".

A special feature of this Editor is that option] will allow you to search for any string, INCLUDING a keyword, if you wish. This is achieved by special procedures which use the table at &8071 in the Electron's ROM to perform a conversion of the keyword in the string that you specify.

PROGRAM DETAILS

The program is in fact a combination of several procedures. We are most grateful to T. Burgess for the first (PROCREPL) and to P. Otto for PROCLISTT, PROCLISTP and PROCLISTF.

The procedures have been combined and are accessed via a menu, to create a single more powerful Editor. However, if you decide that you are interested say, in only one of the procedures, they are easily separated.

LIMITATIONS

The Editor is stored for convenience just below Mode 6 HIMEM. Consequently running a development program which uses a Mode other than 6, with the Editor also in memory, will overwrite the Editor. This is no real problem as it is not necessary to run the user program while the Editor is in situ, and in any case, the Editor is easily reloaded.

```
  10 REM Program UTILED
  20 REM Authors P.OTTO, T.BURGESS, A.
CALCRAFT
  30 REM Version E0.5
  40 REM ELBUG  APRIL 1984
  50 REM Program subject to copyright
  60 :
 100 ON ERROR GOTO 1770
 110 *KEY0PAGE=&5000|MRUN|M
 120 page=&E00:REM Type page in lower
case
 130 DIM T%12,R%12,P%250
 140 MODE6
 150 CLS:PRINT TAB(12,1)"ELBUG EDITOR"
''TAB(15)"-MENU-"
 160 PRINT TAB(5,6)"1..STRING SEARCH..
INCL/KEYWDS/"
 170 PRINT TAB(5,8)"2..STRING REPLACE"
 180 PRINT TAB(5,10)"3..LIST PROCEDURE
S"
 190 PRINT TAB(5,12)"4..LIST FUNCTIONS
"
 200 PRINT TAB(5,14)"5..RETURN TO PROG
RAM BEING EDITED"
 210 PRINT TAB(0,16)"ESCAPE..RETURN TO
 THIS MENU"
 220 PRINT ''"Please select option :";
 230 G$=GET$:IF G$<"1" OR G$>"5" VDU 7
:GOTO 230
 240 PRINT G$'
 250 G=VAL(G$):ON G GOTO 260,270,280,2
90,310
 260 INPUT"SEARCH STRING.."$P%: PROCST
RT: PROCLISTT($P%):GOTO 300
 270 PROCREPL:GOTO 300
 280 PROCLISTP:GOTO 300
```

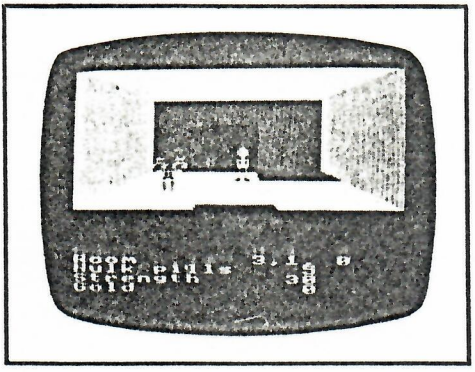
```
  290 PROCLISTF
  300 INPUT'"COMPLETED..PRESS RETURN" $
P%:GOTO150
  310 PAGE=page:END
  320 :
 1000 DEFPROCREPL
 1010 B%=page+1:Q%=B%:INPUT"SEARCH STRI
NG.."$T%:INPUT"REPLACE STRING.."$R% :PR
INT'"THE FOLLOWING LINES ALTERED.."': V
DU14
 1020 IF ?(Q%)=&FF OR LEN($R%)>LEN($T%)
 PRINT"Error, try again":VDU15:ENDPROC
 1030 REPEAT:V%=?(B%+2):FOR X%=1 TO V%:
?(Q%+X%-1)=?(B%+X%-1):NEXT
 1040 REPEAT:IF ?(Q%+2)-4<LEN($T%) LOC=
0:GOTO1080
 1050 $P%="":FOR C%=Q%+3 TO Q%+?(Q%+2)-
2:$P%=$P%+CHR$(?C%):NEXT:LOC=INSTR($P%,
$T%):IF LOC=0 GOTO 1080 ELSE IF $T%=$R%
 PRINT 256*?(Q%)+?(Q%+1):LOC=0:GOTO1080
 1060 FOR C%=1 TO LEN($R%):?(Q%+LOC+1+C
%)=ASC(MID$($R%,C%,1)):NEXT:FOR C%=1 TO
 ?(Q%+2)-LOC-LEN($T%)-2:?(Q%+LOC+LEN($R
%)+C%+1)=?(Q%+LOC+LEN($T%)+C%+1):NEXT
 1070 ?(Q%+2)=?(Q%+2)+LEN($R%)-LEN($T%)
:PRINT 256*?(Q%)+?(Q%+1)
 1080 UNTIL LOC=0:Q%=Q%+?(Q%+2):B%=B%+V
%
 1090 UNTIL (?(B%-1)=&0D AND ?(B%)=&FF)
 OR B%=LOMEM:?(Q%)=&FF
 1100 VDU15:ENDPROC
 1110 :
 1120 DEFPROCLISTT($P%):PROCPROGSCH($P%
,0):PRINT:ENDPROC
 1130 :
 1140 DEFPROCLISTP
 1150 PRINT"Searching for DEFPROC"':PRO
CPROGSCH(CHR$&DD+CHR$&F2,-1)
 1160 PRINT"Searching for DEF PROC"':PR
OCPROGSCH(CHR$&DD+" "+CHR$&F2,-1)
 1170 ENDPROC
 1180 :
 1190 DEFPROCLISTF
 1200 PRINT"Searching for DEFFN"':PROCP
ROGSCH(CHR$&DD+CHR$&A4,-1)
 1210 PRINT"Searching for DEF FN":PROCP
ROGSCH(CHR$&DD+" "+CHR$&A4,-1)
 1220 ENDPROC
 1230 :
 1240 DEFPROCPROGSCH($P%,DISP%)
 1250 K%=LEN($P%):VDU14:A%=@%:@%=8:AST$
=""
 1260 IF DISP%@%=6:AST$="*"
 1270 I%=page-1
 1280 REPEAT:I%=I%+1:IF?I%=P%?0 PROCLOO
P
 1290 UNTIL ?I%=&0D AND I%?1=&FF:VDU15:
PRINT:@%=A%
 1300 ENDPROC
 1310 :
 1320 DEFPROCSTRT
 1330 A%=P%-1:Q%=P%+LEN($P%):Y%=0
 1340 FORN%=1 TO2
 1350 X%=Y%
 1360 REPEAT:A%=A%+1:UNTIL ?A%=&2F OR A
%>=Q%
 1370 IF ?A%=&2F Y%=A%
 1380 NEXT
 1390 IF X%=0 OR X%=Y% ENDPROC
 1400 X%=X%+1:B%=Y%-X%
 1410 PROCTEST
 1420 IF F%=&FF PRINT"INVALID KEYWORD":
ENDPROC
 1430 A%=P%:P%=P%-1
 1440 REPEAT P%=P%+1:UNTIL ?P%=&2F
 1450 ?P%=F%: Z%=P%+1
 1460 REPEAT P%=P%+1:UNTIL ?P%=&2F
 1470 REPEAT: P%=P%+1:?Z%=?P% :Z%=Z%+1:
UNTIL P%=Q%
 1480 P%=M%:$P%=MID$($A%,1,Z%-A%)
 1490 ENDPROC
 1500 :
 1510 DEFPROCTEST
 1520 F%=0
 1530 D%=&8071
 1540 REPEAT
 1550 IF ?D%=?X% PROCCHK
 1560 IF F%=0 REPEAT:D%=D%+1:UNTIL ?D%>
&7F:D%=D%+2
 1570 IF D%>&836B OR ?D%>?X% F%=&FF
 1580 UNTIL F%<>0
 1590 ENDPROC
 1600 :
 1610 DEFPROCCHK
 1620 C%=D% :F%=1
 1630 FOR N%=0 TO B%-1
 1640 IF X%?N% <> ?C% F%=0
 1650 C%=C%+1:NEXT
 1660 IF ?C% < &80 OR C%?1 > &24 AND C%
?1 <> &43 F%=0
 1670 IF F%=1 F%=?C%
 1680 ENDPROC
 1690 :
 1700 DEFPROCLOOP
 1710 C%=1:FOR J%=I%+1 TO I%+K%: IF ?J%
=P%?C% C%=C%+1 ELSE J%=K%+I%
 1720 NEXT:IF C%<K% ENDPROC
 1730 FOR J%=I%-1 TO I%-254 STEP-1
 1740 IF ?(J%-3)=13 PRINT AST$;?(J%-1)+
?(J%-2)*256,;: J%=I%-254:IF DISP%X%=I%+
K%:PRINT" ";:REPEAT:Y%=?X%:PRINT CHR$(Y
%);:X%=X%+1:UNTIL Y%=13 OR Y%=58 OR Y%=
61:PRINT
 1750 NEXT:ENDPROC
 1760 :
 1770 ON ERROR OFF
 1780 IF ERR=17 RUN
 1790 MODE 6
 1800 REPORT:PRINT" at line ";ERL
 1810 END
```

LATEST GAMES REVIEWED

Title : ESCAPE FROM MOONBASE ALPHA
Supplier: Program Power
Price : £7.95
Reviewer: Alan R.Webster
Rating : ****

'Escape from Moonbase Alpha' is a three-dimensional adventure game loosely based on the television series, 'Dr.Who'. The adventure takes place in a series of linked rooms, each of which is represented in a three-dimensional form on the screen when you visit it. This is an unusual and intriguing feature of this game.

The object of the game is to collect as many bags of gold as possible, whilst fighting off the challenge of the various monsters. You start in a room on level zero, and to escape you have to find 'The Doctor' who is down on level seven. The price of freedom is ten bags of gold, so make sure you have plenty of gold before finding the Doctor. Most rooms have various connecting doorways, or stairways, leading to other rooms in the game.



You will sometimes find a 'TARDIS' in a room you visit, and if you can enter the TARDIS quickly you dematerialise and then reappear in a quite different room: This may help or hinder your progress. As you move through the levels, the monsters become stronger and stronger, but at the same time, each bag of gold that you acquire also increases your own strength.

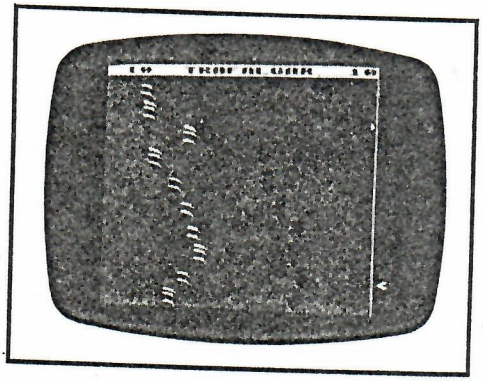
This is a very nice game, with some enjoyable graphics, and makes a nice relaxing change to the "zap 'em" type games. It certainly rates very high in the tables for the best Electron software.

Titles : BUN FUN, SUPER GOLF, TRAFALGAR
Supplier : SQUIRREL SOFTWARE
Prices : £6.50, £7.50, £8.00
Reviewer : Philip Le Grand
Ratings : *, **, ***

The three games reviewed here all come from a relatively new software company called Squirrel Software. The first of the three, BUN FUN, has a rather uninspiring name, and once running on the Electron, turned out to be a rather uninspiring game. You are in a factory which makes buns, and you have been left in charge of placing the icing and nuts on top of the buns. Your pay at the end of the day depends on the number of completed buns you produce and the amount of waste you create. After playing this game a couple of times, I had no desire to continue playing it.

The second game, SUPER GOLF, was an improvement over the last, in that there was more to think about and was enjoyable enough to come back and play it again. The choice of colours used in the program made it difficult, often impossible, to follow the ball (although if you are a keen spectator of golf on television, you may well be accustomed to the fine art of finding the ball!). Up to four players are allowed to play this game, and at the end of every hole, a score card is displayed for each player. The course

consists of the usual eighteen holes, with the option to miss out any hole. It is a pity that this game does not generate sound effects, especially for driving off, and that the ball does not roll on if it lands on the side of the mountain the program often creates for some of the greens.

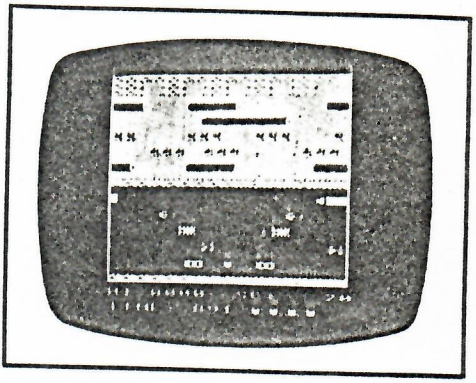


The third and final game, called TRAFALGAR, was the best out of all the games from Squirrel Software. The object of the game is to sink as many of the opposition's ships as possible at the Battle of Trafalgar. The game allows two players or one player and the computer to do battle. Sometimes, the computer seems to have a remarkable aim, especially if it's losing! The graphics were clear and well designed, with interesting sound effects. The visual effects of the ships exploding and sinking were particularly striking.

In conclusion, BUN FUN was not an enjoyable game, and SUPER GOLF did not play as realistic a game of golf as other similar programs on the market. TRAFALGAR is the best game so far from Squirrel Software, though if they want to compete seriously with some of the larger software producers, they will have to increase the standard of their games software quite considerably.

Title : CROAKER
Supplier : Program Power
Price : £7.95
Reviewer : Mike Williams
Rating : ****

Croaker is another version of the game also known as Frogger. The game itself is delightfully simple to understand. You have to control the movement of a frog who has to reach his home by crossing a busy five lane motorway, and then a polluted river by jumping on to floating logs and the backs of turtles. The visual presentation is extremely good with multi-coloured road vehicles of all sizes and very realistic logs (rather less realistic turtles).



The journey is initially not too hazardous, but the game requires you to keep on trying. After the first screen, the vehicles are closer together, making the journey more dangerous, while logs occasionally turn out to be well disguised and not very friendly crocodiles. Even the placid turtles can make life dangerous by diving suddenly, to leave you floundering in the polluted river. The program includes an optional musical accompaniment which you either like or hate. This is a first class game for the Electron and good value at £7.95.

USING BBC MICRO PROGRAMS ON AN ELECTRON (Part 2)

by David Graham

This month, David Graham continues his series on converting BBC micro programs to run on the Electron, by looking at the differences in the use of the SOUND and ENVELOPE commands.

Sounds are produced on the Electron and the BBC micro by using the two commands SOUND and ENVELOPE. Their implementation on the Electron is more limited than on the BBC micro, as suggested last month. In broad terms there is only one noise and one tone channel on the Electron (one noise and three tone on the Beeb), and amplitudes may only take one of two values on the Electron - ie. on or off (the Beeb has sixteen selectable sound levels). The Electron's ENVELOPE allows only pitch to be varied, while the Beeb's gives both pitch and amplitude variation.

Programs on the Beeb with sound effects usually need some modification before they will work satisfactorily on the Electron from an acoustic point of view. Because there is only one tone channel on the Electron, it is not usually worth attempting to convert programs whose main function is to generate music, since the result will be generally unrewarding. But otherwise, it us usually possible to produce replacement sound effects - though the generation of a good explosion, or gunshot is not possible because of the restriction on amplitude variation.

In seeking to convert Beeb sounds to Electron sounds, it is probably best to consider the SOUND and ENVELOPE commands separately. We will look at the SOUND command first.

SOUND

The SOUND command has four parameters, Q, A, P and D. You may like to refer briefly to page 116 of the User Guide if you are unfamiliar with the SOUND call. To produce a sound, try typing

```
SOUND 1,-15,100,40
```

In this case Q=1, A=-15, P=100 and D=40

Q selects the sound channel number
A switches the sound on or off or selects ENVELOPE.
P selects the pitch
D selects the duration

If you run a sound generating routine for the Beeb on your Electron, and the program has no associated ENVELOPE, then the result will be as follows.

1. Any values of amplitude chosen in the Beeb SOUND command (from -15 to -1) will give full volume on the Electron. A zero will give "off" on either machine. So there is no real problem here.

2. If the Beeb uses more than one sound channel at the same time, then the Electron will lose sounds, simply playing the last in the sequence. Thus with the following two lines of code:

```
100 SOUND 1, -15, 100,100
110 SOUND 2, -10, 50, 10
```

the Beeb would begin both sounds simultaneously, and the first, which is 10 times longer than the second, would play for 5 seconds. On the Electron, the second would overwrite the first before it was played, and you would just hear the half second 'blip' generated by line 110. If a program requires a longer sound in such a case, the simple solution is to delete the shorter one if it is listed after the longer one. If, alternatively, you wanted to hear the sounds in sequence, then you should change the '2' in line 110 to '1'.

3. Unfortunately, it is not possible to get the noise channel on the Electron to generate sounds simultaneously with the tone channel. The result is similar to 2 above. The last command in any group is the only one heard. Complex sound effects on the Beeb will thus produce various results depending on the sequence of commands. Thus on the Beeb the following:

```
100 SOUND 1,-15,100,100
110 SOUND 0,-15,100,10
```

would produce a long tone accompanied by a short 'blip' of noise at the start. On the Electron, you only hear the blip. If you reverse the order of the commands, there will be no change on the Beeb, but on the Electron you will hear only the long tone. The noise will be lost.

4. Finally, the production of noise (opposed to pure tones) on the Electron is disrupted by other work which the CPU is forced to do (Central Processor Unit - the 6502 chip at the heart of the Electron and the Beeb). This does not happen on the Beeb, and can be used to good effect on the Electron. For example, try the following:

```
100 SOUND 0,-15,100,100
```

Now add:

```
110 FOR A=1 TO 3000:NEXT
```

and run it again. You will hear that the first half of the noise is warbled because the effective output frequency is drastically cut as the CPU attends to this second line of Basic.

ENVELOPE

The only thing that causes a problem with the ENVELOPE command is the lack of amplitude variation. Contrary to what I said last month, the Electron possesses a full set of 16 Envelopes, making life a good deal easier than I had envisaged.

All the effects mentioned under 'SOUND' above are retained when envelopes are used, and the same techniques of restoring sounds should be used.

The missing segment of the ENVELOPE command (the last 6 parameters in fact) simply means that the Beeb programs which use this to vary the volume of the sound or to turn it on or off may need modifying.

For example, try the following:

```
10 ENVELOPE 2,1,0,0,0,6,3,3,30,-4,0,-5,
120,80
20 SOUND 2,2,100,7
```

On the Beeb, this gives a note of constant pitch which dies away slowly. On the Electron, it stays at full volume. The same occurs with explosion sounds. Try replacing line 20 above with:

```
20 SOUND 0,2,6,15
```

On the Beeb, this causes the sound of an explosion or gun shot, but on the Electron it just sounds like a burst of noise.

In fact, one can do very little to reproduce such an effect, and the best approach in this case is probably to replace the sound with another of your choice, possibly by trial and error.

One substitute might be to try to use the facility for varying the pitch of white noise. As an example of this, try the following:

```
10 ENVELOPE 1,3,4,4,10,20,10,127,0,0,0,
0,0,0
20 SOUND 0,1,100,100
```

Next month we shall look at ways of getting the maximum speed and response out of the Electron. Because of the way the Electron is designed, compared with the Beeb, it is necessary to consider a variety of time saving techniques if we are to achieve comparable speed of operation in many cases.

HINTS HINTS HINTS HINTS HINTS

TIME DELAY

A good way to put a time delay in a program is to use the INKEY function. For example, X=INKEY(200) will give a two second delay (the number in brackets is the time in centiseconds). Note that this delay can be cut short by pressing a key, which can be useful when testing.

ELEVATOR MANIA

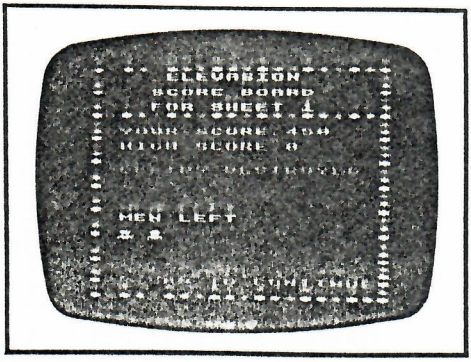
by D. J. Pilling

If you enjoyed the Block Blitz game last month, then you will be positively dazzled by 'Elevator Mania', a super fast action game by the same author. This is one of the fastest games written in Basic that we have seen for a long time and will provide you with hours of adrenalin sapping enjoyment in return for the effort of typing it into your micro.

Elevator Mania is a one player game, in which you control a man running left and right along a vast network of corridors, or going up and down in a bewildering maze of lifts. The object of the game is to kill as many 'droids' as possible by luring them under the mines which hang from the roofs of corridors. When this happens, the droids are immoblised for a short time, allowing you to run underneath and kill them. If you fail to kill the droid, it will be released and come chasing after you once again. If a second droid passes a droid that has already run into a mine, both droids will be released.

The mines also halt the progress of the man for a few seconds - time enough unfortunately for the droids to catch him in most cases.

After five screens of play, the mines will start to be replaced by 'blue cloners'. These have the alarming effect of duplicating any droid that passes underneath.



The game is controlled from the keyboard with 'Z' and 'X' moving the man left and right. Although the lifts move up and down quite automatically, you can manually control the lifts using '*' for up and '?' for down. You will also find that if the man disappears off the screen at one side, he will re-appear on the same level at the other side of the screen. The droids cannot follow you in this way.

If you find the game too fast for you initially, you can slow it down by adding an extra delay line to the program. For example:

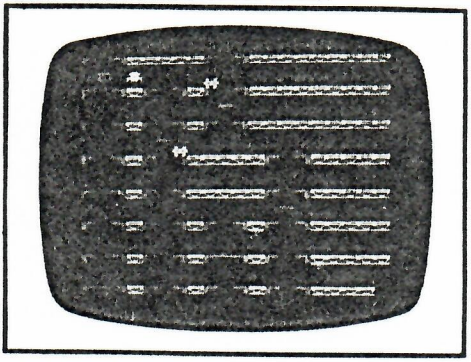
```
2345 TIME=0:REPEAT UNTIL TIME>5
```

The larger the value at the end of the line the greater the delay and the slower the game.

This is a quite outstanding game, particularly when you consider that it is written in Basic and not machine code. Despite its length, which we have kept as reasonable as possible, we are sure you will agree with us that this is a truly outstanding game.

```
  10 REM PROGRAM ELEVATOR
  20 REM AUTHOR  D.J.Pilling
  30 REM VERSION E0.2
  40 REM ELBUG   APRIL 1984
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 ON ERROR GOTO 240
 110 MODE6:VDU23,1,0;0;0;0;
 120 PROCFP
 130 MODE5:PROCCHAR
 140 DIM A$(1),B$(1),C$(1)
 150 DIMX%(6),Y%(6),U%(6),L%(6),D%(6),
A%(6),B%(6),Z%(6),W%(6),I%(6),P%(6),U$(
6),D$(6),L$(1),R$(1),M$(1),A%330
 160 PROCS
 170 REPEAT:PROCST
 180 REPEAT:PROCMA
 190 REPEAT:PROCL:PROCD:PROCM:UNTILOV%
 200 PROCG
 210 UNTIL ME%=0:PROCNXG
 220 UNTIL FALSE
 230 :
```

```
  240 ON ERROR OFF
  250 MODE 6:IF ERR=17 END
  260 REPORT:PRINT " at line ";ERL
  270 END
  280 :
 1000 DEFPROCMA
 1010 CLS:CF%=FALSE:VDU23,1,0;0;0;0;
 1020 VDU19,3,6,0,0,0
 1030 PROCRM:PROCGM
 1040 COLOUR2
 1050 FORI%=3TO27STEP4:PRINTTAB(0,I%)ST
RING$(20,CHR$240);:NEXT:PRINTTAB(0,31)S
TRING$(19,CHR$240);
 1060 FORI%=A%TOA%+287STEP21:$I%=STRING
$(20,CHR$7)+CHR$0:NEXT
 1070 A%?166=0:COLOUR1
 1080 FORI%=1TO6:FORJ%=U%(I%)TOL%(I%)
 1090 IF (J%+1)MOD4=0 COLOUR1:PRINTTAB(
X%(I%)-1,J%)CHR$252LB$CHR$253;:COLOUR1
ELSEPRINTTAB(X%(I%),J%)LB$;
 1100 K%=((J%+1)DIV4-1)*21+A%+X%(I%)
 1110 ?K%=I%:NEXT:I%(I%)=0
 1120 PRINTTAB(X%(I%),Y%(I%))L$;:NEXT
 1130 COLOUR3
 1140 FORI%=1TOND%
 1150 PROCSA
 1160 ?H%=8:PRINTTAB(J%,K%*4+2)CHR$246;
 1170 A%(I%)=J%:B%(I%)=K%*4+2
 1180 Z%(I%)=A%+K%*21:W%(I%)=0
 1190 NEXT
 1200 FORI%=1TONM%:PROCSM:NEXT
```



```
 1210 IFNC%=0ENDPROC
 1220 COLOUR3
 1230 FORI%=1TONC%
 1240 PROCSA
 1250 ?H%=10:PRINTTAB(J%,K%*4)CHR$242;
 1260 NEXT
 1270 ENDPROC
 1280 :
 1290 DEFPROCSA
 1300 J%=RND(18):K%=RND(7):H%=A%+J%+K%*
21
 1310 IF?H%<>7OR?(H%-21)<>7OR?(H%+1)<>7
OR?(H%-1)<>7OR?(H%-20)<>7OR?(H%-22)<>7
GOTO1300
 1320 ENDPROC
 1330 :
 1340 DEFPROCSM:PROCSA:?H%=9:COLOUR1:PR
INTTAB(J%,K%*4)CHR$242;:COLOUR3:ENDPROC
 1350 :
 1360 DEFPROCRM
 1370 X%=0:Y%=1:M%=1:Z%=A%:W%=FALSE
 1380 ENDPROC
 1390 :
 1400 DEFPROCGM
 1410 X%(1)=FNRX:U%(1)=3:L%(1)=U%(1)+4+
RND(4)*4
 1420 X%(2)=FNRX:IFX%(2)=X%(1)GOTO1420
 1430 L%(2)=31:U%(2)=L%(1)-RND(2)*4
 1440 FORI%=3TO6
 1450 X%(I%)=FNRX:K%=0
 1460 FORJ%=1TOI%-1
 1470 IFX%(I%)<>X%(J%)GOTO1490
 1480 IFK%=0K%=J%ELSEK%=-1
 1490 NEXT
 1500 IFK%=0PROCNL:GOTO1550
 1510 IFK%=-1GOTO1450
 1520 U%(I%)=U%(K%)-3:L%(I%)=31-L%(K%)
 1530 IFU%(I%)<12ANDL%(I%)<12GOTO1450
 1540 IFU%(I%)<L%(I%)PROCNB ELSE PROCNT
 1550 NEXT
 1560 FORI%=1TO6:U$(I%)=MD$:D$(I%)=MD$:
U$(I%)=LU$:D$(I%)=LD$:D%(I%)=RND(2)-1:Y
%(I%)=(U%(I%)+L%(I%))/2:NEXT
 1570 ENDPROC
 1580 :
 1590 DEFPROCNL
 1600 U%(I%)=-1+RND(5)*4:L%(I%)=U%(I%)+
8+4*RND(3):IFL%(I%)>31L%(I%)=31
 1610 ENDPROC
 1620 :
 1630 DEFPROCNT
 1640 U%(I%)=-1+RND(2)*4:L%(I%)=U%(I%)+
4+RND(2)*4
 1650 IFL%(I%)>=U%(K%)GOTO1640
 1660 ENDPROC
 1670 :
 1680 DEFPROCNB
 1690 L%(I%)=35-RND(2)*4:U%(I%)=L%(I%)-
4-4*RND(2)
 1700 IFU%(I%)<=L%(K%)GOTO1690
 1710 ENDPROC
 1720 :
 1730 DEFFNRX=4*RND(5)-3
 1740 :
 1750 DEFPROCL
 1760 COLOUR1
 1770 FORI%=1TO6
 1780 IFD%(I%)GOTO1810
 1790 PRINTTAB(X%(I%),Y%(I%))U$(I%);:Y%
(I%)=Y%(I%)-2:IFY%(I%)=U%(I%)D%(I%)=TRU
E
 1800 NEXT:ENDPROC
 1810 PRINTTAB(X%(I%),Y%(I%))D$(I%);:Y%
(I%)=Y%(I%)+2:IFY%(I%)=L%(I%)D%(I%)=FAL
SE
```

```
 1820 NEXT:ENDPROC
 1830 :
 1840 DEFPROCCHAR
 1850 VDU23,240,255,129,255,129,66,36,2
4,255
 1860 VDU23,241,255,255,255,0,0,0,0,0
 1870 VDU23,242,255,255,60,24,24,90,66,
66
 1880 VDU23,243,60,90,126,60,36,126,255
,189
 1890 VDU23,244,189,189,60,36,36,36,100
,6
 1900 VDU23,245,189,189,60,36,36,36,38,
96
 1910 VDU23,246,195,66,126,219,219,126,
66,195
 1920 VDU23,247,66,66,231,66,66,231,66,
66
 1930 VDU23,248,204,51,204,51,204,51,20
4,51
 1940 VDU23,252,255,255,255,224,224,224
,224,224
 1950 VDU23,253,255,255,255,7,7,7,7,7
 1960 VDU23,224,28,48,62,20,28,60,126,1
89
 1970 VDU23,225,189,60,24,24,56,104,200
,76
 1980 VDU23,226,189,60,24,24,28,23,18,2
4
 1990 VDU23,227,189,60,24,24,28,23,18,4
8
 2000 VDU23,228,189,60,24,24,56,104,200
,88
 2010 VDU23,229,28,6,62,20,28,60,126,18
9
 2020 VDU23,230,102,66,126,219,219,126,
66,102
 2030 ENVELOPE1,1,6,0,-6,200,100,200,10
0,2,0,-1,120,110
 2040 ENDPROC
 2050 :
 2060 DEFPROCS
 2070 TE$=CHR$10+CHR$8+CHR$8
 2080 ET$=CHR$8+CHR$10:EE$=CHR$8+CHR$11
 2090 EG$=CHR$11+CHR$8:R$=CHR$17+CHR$1
 2100 Y$=CHR$17+CHR$2:W$=CHR$17+CHR$3
 2110 B$=CHR$32+CHR$10+CHR$8+CHR$32
 2120 L$=CHR$241:LB$=CHR$32
 2130 LU$=LB$+CHR$11+EG$+L$
 2140 LD$=LB$+CHR$10+CHR$10+CHR$8+L$
 2150 MU$=LB$+EG$+LB$+EE$+L$+EE$+R$+CHR
$244+Y$+EE$+CHR$243+R$
 2160 MD$=LB$+Y$+EE$+CHR$11+LB$+ET$+LB$
+ET$+CHR$243+ET$+R$+CHR$244+ET$+R$+L$
 2170 TP$=Y$+CHR$8+CHR$229+LB$+TE$+R$:L
$(0)=TP$+CHR$227+LB$:L$(1)=TP$+CHR$228+
LB$
 2180 TP$=Y$+LB$+CHR$224+TE$+LB$+R$:R$(
0)=TP$+CHR$225:R$(1)=TP$+CHR$226
 2190 TP$=Y$+CHR$243+CHR$10+CHR$8+R$:M$
(0)=TP$+CHR$244:M$(1)=TP$+CHR$245
 2200 A$(0)=CHR$8+CHR$246+LB$
 2210 A$(1)=CHR$8+CHR$230+LB$
 2220 B$(0)=LB$+CHR$246
 2230 B$(1)=LB$+CHR$230
 2240 C$(0)=CHR$246:C$(1)=CHR$230
 2250 P$=A$(0):S$=B$(0):C$=C$(0)
 2260 DU$=LB$+EG$+LB$+EE$+L$+EE$+W$+CHR
$246+R$
 2270 DD$=LB$+W$+EE$+LB$+ET$+CHR$10+CHR
$246+ET$+R$+L$
 2280 CH$=R$+CHR$42+Y$+CHR$42+W$+CHR$42
 2290 CV$=Y$+CHR$42+ET$+R$+CHR$42+ET$+W
$+CHR$42+ET$
 2300 ?A%=0:A%=A%+1
 2310 HSC%=0:N$="JOE ZERO"
 2320 ENDPROC
 2330 :
 2340 DEFPROCM
 2350 IFW%GOTO2440
 2360 IFM% M%=0ELSEM%=1
 2370 P%=Z%+X%:IF?P%>7PROCE2:ENDPROC
 2380 IFINKEY-67GOTO2400ELSEIFINKEY-98G
OTO2420
 2390 PRINTTAB(X%,Y%)M$(M%):ENDPROC
 2400 IFP%?1>6PRINTTAB(X%,Y%)R$(M%):X%=
X%+1ELSEL%=P%?1:PROCEN
 2410 ENDPROC
 2420 IFP%?-1>6PRINTTAB(X%,Y%)L$(M%):X%
=X%-1ELSEL%=P%?-1:PROCEN
 2430 ENDPROC
 2440 IFW%<0GOTO2510
 2450 IFINKEY-67 TX%=1:PROCEX:ENDPROC
 2460 IFINKEY-98 TX%=-1:PROCEX:ENDPROC
 2470 Y%=Y%(L%)
 2480 IFINKEY-73D%(L%)=0 ELSEIFINKEY-10
5D%(L%)=1 ELSEENDPROC
 2490 IFY%(L%)=U%(L%)D%(L%)=TRUE ELSEIF
Y%(L%)=L%(L%)D%(L%)=FALSE
 2500 ENDPROC
 2510 W%=W%+1:IFW%=0:PROCSM
 2520 ENDPROC
 2530 :
 2540 DEFPROCEN
 2550 IFL%=0GOTO2620
 2560 IFY%(L%)<>Y%+2ORI%(L%)PRINTTAB(X%
,Y%)M$(M%):ENDPROC
 2570 SOUND1,1,100,2:PRINTTAB(X%,Y%)B$
 2580 U$(L%)=MU$:D$(L%)=MD$:X%=X%(L%)
 2590 PRINTTAB(X%,Y%)M$(M%)
 2600 W%=1:I%(L%)=9
 2610 ENDPROC
 2620 IFX%=0PRINTTAB(0,Y%)B$:X%=19+(Y%=
29):PRINTTAB(X%,Y%)M$(M%):ENDPROC
 2630 IFX%>17 PRINTTAB(X%,Y%)B$:X%=0:PR
INTTAB(X%,Y%)M$(M%):ENDPROC
 2640 PRINTTAB(X%,Y%)M$(M%)
 2650 ENDPROC
 2660 :
 2670 DEFPROCEX
 2680 IF(Y%(L%)+1)MOD4=0ELSEENDPROC
 2690 SOUND1,1,200,2
```

```
 2700 U$(L%)=LU$:D$(L%)=LD$:W%=FALSE
 2710 I%(L%)=0
 2720 Y%=Y%(L%)-2:Z%=((Y%(L%)+1)DIV4-1)
*21+A%
 2730 PRINTTAB(X%,Y%)B$:X%=X%+TX%
 2740 PRINTTAB(X%,Y%)M$(M%):ENDPROC
 2750 :
 2760 DEFPROCD
 2770 IFE% E%=0ELSEE%=1
 2780 P$=A$(E%):S$=B$(E%):C$=C$(E%)
 2790 COLOUR3
 2800 FORD%=1TO5
 2810 IFD%>ND%:PROCDT(0):NEXT:ENDPROC
 2820 IFW%(D%)GOTO2900
 2830 U%=A%(D%):V%=Z%(D%)+U%:S%=B%(D%)
 2840 IFS%=Y%+1P%(D%)=U%<X%:IFU%=X%NEXT
:ENDPROC
 2850 IFP%(D%)GOTO2880
 2860 IFV%?-1>6ELSET%=V%?-1:PROCDE:NEXT
:ENDPROC
 2870 ?V%=7:PRINTTAB(U%,S%)P$:A%(D%)=U%
-1:IFV%?-1<9V%?-1=8:NEXT:ENDPROC ELSEPR
OCE1M:NEXT:ENDPROC
 2880 IFV%?1>6ELSET%=V%?1:PROCDE:NEXT:E
NDPROC
 2890 ?V%=7:PRINTTAB(U%,S%)S$:A%(D%)=U%
+1:IFV%?1<9V%?1=8:NEXT:ENDPROC ELSEPROC
E1P:NEXT:ENDPROC
 2900 IFW%(D%)<0GOTO2980
 2910 R%=Y%-Y%(W%(D%))+2
 2920 IFR%=0PROCDX:NEXT:ENDPROC
 2930 IFR%<0GOTO2960
 2940 IFNOTD%(W%(D%))PROCDX
 2950 NEXT:ENDPROC
 2960 IFD%(W%(D%))PROCDX
 2970 NEXT:ENDPROC
 2980 W%(D%)=W%(D%)+1
 2990 IFW%(D%)=0 PROCRE:SOUND3,1,200,10
 3000 NEXT:ENDPROC
 3010 :
 3020 DEFPROCDE
 3030 IFS%=Y%(T%)-1ELSEIFT%=0P%(D%)=NOT
P%(D%):ENDPROC ELSEPRINTTAB(U%,S%)C$:EN
DPROC
 3040 IFS%+1=U%(T%)ORS%+1=L%(T%)GOTO309
0
 3050 IFY%<=S%GOTO3080
 3060 IFNOTD%(T%)ENDPROC
 3070 GOTO3090
 3080 IFD%(T%)ENDPROC
 3090 IFI%(T%)ENDPROC
 3100 ?V%=7
 3110 SOUND1,1,100,2:PRINTTAB(U%,S%)LB$
 3120 U$(T%)=DU$:D$(T%)=DD$:A%(D%)=X%(T
%)
 3130 PRINTTAB(A%(D%),S%)C$
 3140 W%(D%)=T%:I%(T%)=D%
 3150 ENDPROC
 3160 :
 3170 DEFPROCDX
 3180 IF(Y%(W%(D%))+1)MOD4=0ELSEENDPROC
 3190 SOUND1,3,200,2
 3200 T%=W%(D%):W%(D%)=0:I%(T%)=0
 3210 U$(T%)=LU$:D$(T%)=LD$:B%(D%)=Y%(T
%)-1
 3220 PRINTTAB(A%(D%),B%(D%))LB$
 3230 IFY%=B%(D%)-1 P%(D%)=A%(D%)<X%
 3240 IFP%(D%) A%(D%)=A%(D%)+1ELSEA%(D%
)=A%(D%)-1
 3250 PRINTTAB(A%(D%),B%(D%))C$(E%)
 3260 Z%(D%)=((Y%(T%)+1)DIV4-1)*21+A%:?
(Z%(D%)+A%(D%))=8
 3270 ENDPROC
 3280 :
 3290 DEFPROCE1M
 3300 U%=U%-1:V%=V%-1:GOTO3340
 3310 :
 3320 DEFPROCE1P
 3330 U%=U%+1:V%=V%+1
 3340 IF?V%=10CF%=TRUE
 3350 IF?V%>10 EX%=D%:D%=?V%-10:SOUND1,
1,200,10:PROCRE:D%=EX%:A%(D%)=A%(D%)+1:
?V%=8:ENDPROC
 3360 ?V%=10+D%:SOUND1,1,200,10:SOUND1,
1,200,10:SOUND2,2,200,30
 3370 FOREX%=1TO36:COLOUREX%MOD4
 3380 PRINTTAB(U%,B%(D%));:VDU246,8,11,
247,8,11,242
 3390 NEXT
 3400 IFCF%PROCCL:ENDPROC
 3410 COLOUR2:PRINTTAB(U%,B%(D%)-2)CHR$
246:COLOUR3:W%(D%)=-50
 3420 ENDPROC
 3430 :
 3440 DEFPROCEA:IFD%=ND%GOTO3460ELSEA%(
D%)=A%(ND%):B%(D%)=B%(ND%):W%(D%)=W%(ND
%):Z%(D%)=Z%(ND%)
 3450 IFW%(ND%)<0 ?(Z%(ND%)+A%(ND%))=10
+D%
 3460 ND%=ND%-1:D%=9:DD%=DD%+1:IFND%=0O
V%=TRUE
 3470 ENDPROC
 3480 :
 3490 DEFPROCRE:PRINTTAB(A%(D%),B%(D%)-
2)LB$;:VDU8,10,10,246:W%(D%)=0:PROCSM:E
NDPROC
 3500 :
 3510 DEFPROCER:D%=?P%-10:?P%=7:COLOUR1
:PRINTTAB(A%(D%),B%(D%)-2)CHR$246:COLOU
R3:PROCEA:SOUND0,1,6,20:ENDPROC
 3520 :
 3530 DEFPROCE2
 3540 IF?P%>10PROCER:ENDPROC
 3550 IF?P%>8W%=-20:?P%=7:PRINTTAB(X%,Y
%-1)LB$:SOUND1,1,200,20:ENDPROC
 3560 SOUND1,1,200,20:PROCDT(25)
 3570 PRINTTAB(X%,Y%)CHR$248:PRINTTAB(X
%,Y%+1)CHR$248:PROCDT(35):PRINTTAB(X%,Y
%)B$
 3580 ME%=ME%-1:DD%=DD%+1:OV%=TRUE
```

```
 3590 ENDPROC
 3600 :
 3610 DEFPROCDT(I%):T=TIME:REPEATUNTILT
IME>T+I%:ENDPROC
 3620 :
 3630 DEFPROCG
 3640 CLS:SC%=SC%+250+200*DD%
 3650 IFSC%DIV2000<>BC% ME%=ME%+1
 3660 IFME%=0ENDPROC
 3670 VDU19,3,2,0,0,0
 3680 PRINTTAB(1,1)STRING$(6,CH$)TAB(0,
1)STRING$(10,CV$)TAB(19,1)STRING$(10,CV
$)TAB(0,30)Y$+CHR$42+STRING$(6,CH$)+Y$+
CHR$42;
 3690 PRINTTAB(5,2)Y$+"ELEVASION"TAB(4,
4)"SCORE BOARD"TAB(4,6)"FOR SHEET ";G%;
TAB(1,7)STRING$(6,CH$);
 3700 PRINTTAB(2,9)"YOUR SCORE ";SC%
 3710 PRINTTAB(2,11)"HIGH SCORE ";HSC%
 3720 PRINTTAB(2,14)R$"DROIDS DESTROYED
"
 3730 X%=DD%:IFX%>8 X%=8
 3740 PRINTTAB(2,16)STRING$(X%,CHR$246+
CHR$9)
 3750 PRINTTAB(2,20)Y$"MEN LEFT"
 3760 X%=ME%:IFX%>8 X%=8
 3770 PRINTTAB(2,22)STRING$(X%,M$(0)+CH
R$11+CHR$9);
 3780 X%=1
 3790 IFSC%DIV2000<>BC%ANDX%=1PRINTTAB(
2,25)Y$"BONUS MAN" ELSEPRINTTAB(2,25)SP
C(10)
 3800 PRINTTAB(1,28)R$"RETURN TO CONTIN
UE"
 3810 IFINKEY-74GOTO3830
 3820 IFTIME MOD15=0 X%=X%EOR1:GOTO3790
 ELSEGOTO3810
 3830 VDU19,3,6,0,0,0:OV%=FALSE
 3840 G%=G%+1:DD%=0:BC%=SC%DIV2000
 3850 IFG%<=5GOTO3890
 3860 ND%=G%-5:IFND%>5ND%=5
 3870 NC%=G%-5:IFNC%>8NC%=8
 3880 NM%=2+ND%:ENDPROC
 3890 ND%=G%:NM%=2+ND%
 3900 ENDPROC
 3910 :
 3920 DEFPROCST
 3930 G%=1:ND%=1:NM%=3:NC%=0:DD%=0:ME%=
3:OV%=FALSE:SC%=0:BC%=0
 3940 ENDPROC
 3950 :
 3960 DEFPROCCL
 3970 CF%=FALSE:SOUND1,1,200,20
 3980 IFND%=5ENDPROC
 3990 ND%=ND%+1:PROCSM
 4000 A%(ND%)=A%(D%)+1:B%(ND%)=B%(D%):Z
%(ND%)=Z%(D%):W%(ND%)=0
 4010 ENDPROC
 4020 :
 4030 DEFPROCNXG
 4040 CLS:VDU19,3,2,0,0,0
 4050 PRINTTAB(1,1)STRING$(6,CH$)TAB(0,
1)STRING$(10,CV$)TAB(19,1)STRING$(10,CV
$)TAB(0,30)Y$+CHR$42+STRING$(6,CH$)+Y$+
CHR$42;
 4060 PRINTTAB(5,2)Y$+"ELEVASION"TAB(4,
4)"GAME   OVER";TAB(1,5)STRING$(6,CH$);
 4070 PRINTTAB(3,7)"THE HISCORE WAS"TAB
(8,9);HSC% TAB(3,11)"BY "N$
 4080 PRINTTAB(3,14)Y$"YOUR SCORE WAS "
TAB(8,16);SC%
 4090 IFSC%<=HSC% GOTO4200
 4100 PRINTTAB(3,18)"THE NEW HISCORE"
 4110 PRINTTAB(3,21)R$"ENTER YOUR NAME"
 4120 *FX21,0
 4130 PRINTTAB(3,23)"-->"Y$;
 4140 N$="":REPEAT IK=GET
 4150 IFIK=13GOTO4180
 4160 IFIK=127ANDLENN$<>0PRINTCHR$IK;:N
$=LEFT$(N$,LENN$-1):GOTO4180 ELSEIFIK=1
27 GOTO4180
 4170 N$=N$+CHR$IK:PRINTCHR$IK;
 4180 UNTIL IK=13OR LENN$=12
 4190 HSC%=SC%
 4200 PRINTTAB(4,26)R$"PRESS RETURN"TAB
(2,28)"FOR ANOTHER GAME";:PROCDT(100):R
EPEAT UNTILINKEY-74
 4210 ENDPROC
 4220 :
 4230 DEFPROCFP
 4240 CLS:PRINTTAB(13,3)"ELEVASION"
 4250 PRINTTAB(16,12);"by"
 4260 PRINTTAB(12,14)"D.J.Pilling "
 4270 PRINTTAB(6,20)"Press any key to s
tart."
 4280 G=GET:CLS
 4290 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS

FREEZING A PROGRAM IN BASIC - Peter Corlett

To make a Basic program stop when 'F' is pressed use:

```
IF INKEY$(0)=70 REPEAT UNTIL GET$="F"
```

This line must be inserted in the main program loop and will cause the program to halt when 'F' is pressed. The program will continue when 'F' is pressed again. This should be inserted as a line in the middle of the main loop of the game.

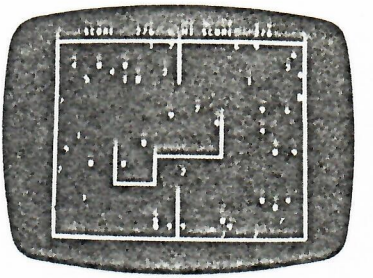
BEEBUGSOFT

Snake

Snake is a very addictive, fast moving arcade game. The purpose of the game is to direct the snake around the screen, eating up the randomly placed objects. With every bite that it takes, the tail grows longer and the snake moves faster. Just to complicate things, there are also certain scattered obstacles which must be avoided.

To even complete the first screen requires skill and concentration. Further frames use different shaped playing areas, which as well as adding variety, require much greater skill to complete.

This game is much more exciting than many games of a similar nature, and once played, proves to be surprisingly compulsive.



Snake costs £4.00 inc VAT. Please add 50p post & packing.
Overseas orders: £5.50 inc post & packing - VAT not charged.
Send order with membership number to:
BEEBUGSOFT, P.O. Box 109, High Wycombe, Bucks

THE BEST OF ELBUG

Many of the best programs published in ELBUG have been collected together and published by Penguin Books under the name "Games and other programs for the Acorn Electron" at £3.95. This book is part of the Penguin Acorn Computer Library and at present there is just one other title available though others are planned.

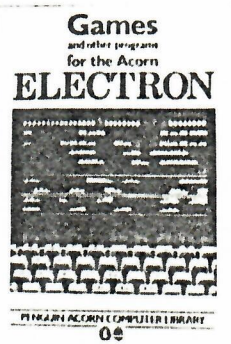
There are 20 programs in all in four different categories:

Action Games
- Munch-Man
- Mars Lander
- Invasion
- Robot Attack
- Hedgehog

Thought games
- Higher/Lower
- Five-Dice
- Life
- Anagrams
- Return of the Diamond

Visual Displays
- Union Jack
- Square Dance
- Ellipto
- Screenplay
- 3-D Rotation

Utilities
- Sound Wizard
- Bad Program Lister
- 3-D Lettering
- Bad Program Rescue
- Double Height Text



All 20 programs are now available on cassette from our software address (in High Wycombe) price £7 to members and £9 to non-members, plus 50p post & packing in either case.

BACK ISSUES AND SUBSCRIPTIONS

BACK ISSUES (Members only)

All back issues will be kept in print (from November 1983). Send 90p per issue PLUS an A5 SAE to the subscriptions address. Back copies of BEEBUG are available to ELBUG members at this same price. This offer is for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the advertising supplements are not supplied with back issues.

Subscription and Software Address

ELBUG
PO BOX 109
High Wycombe
Bucks

SUBSCRIPTIONS

Send all applications for membership, and subscription queries to the subscriptions address.

MEMBERSHIP COSTS:

U.K.
£5.90 for 6 months (5 issues)
£9.90 for 1 year (10 issues)
Eire and Europe
Membership £16 for one year.
Middle East £19
Americas and Africa £21
Elsewhere £23
Payments in Sterling preferred.

SOFTWARE (Members only)

This is available from the software address.

MAGAZINE CONTRIBUTIONS AND TECHNICAL QUERIES

Please send all contributions and technical queries to the editorial address opposite. All contributions published in the magazine will be paid for at the rate of £25 per page.

We will also pay £10 for the best Hint or Tip that we publish, and £5 to the next best. Please send all editorial material to the editorial address opposite. If you require a reply it is essential to quote your membership number and enclose an SAE.

Editorial Address

ELBUG
PO Box 50
St Albans
Herts

ELBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Editor: Mike Williams.
Technical Editor: Philip Le Grand. Production Editor: Phyllida Vanstone.
Technical Assistants Alan Webster and David Fell.
Managing Editor: Lee Calcraft.
Thanks are due to David Graham, Sheridan Williams, and Adrian Calcraft for assistance with this issue.

All rights reserved. No part of this publication may be reproduced without prior written permission of the Publisher. The Publisher cannot accept any responsibility, whatsoever, for errors in articles, programs, or advertisements published. The opinions expressed on the pages of this journal are those of the authors and do not necessarily represent those of the Publisher, BEEBUG Publications Limited.
BEEBUG Publications LTD (c) April 1984.

Printed in England by Staples Printers St Albans Limited at The Priory Press.